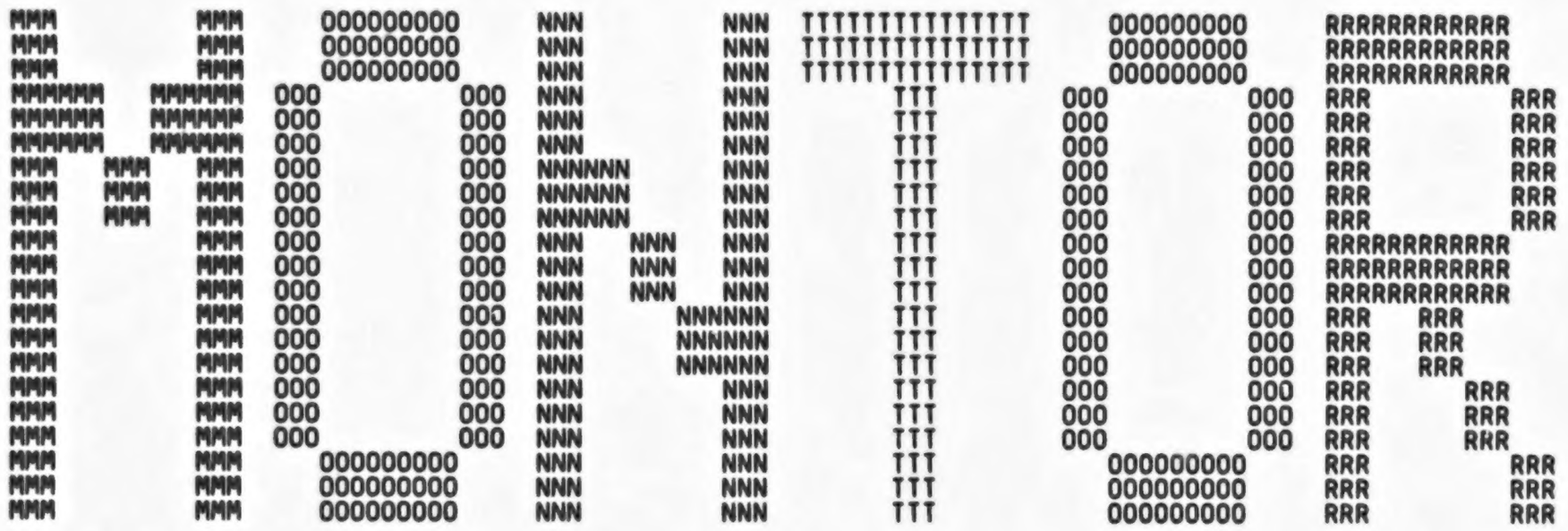
MMM MMM 000000000 NNN NNN TTTTTTTTTTTTTTTTT 000000000 RRRRRRRRRRR
MMM MMM 000000000 NNN NNN TTTTTTTTTTTTTTTTT 000000000 RRRRRRRRRRR
MMM MMM 000000000 NNN NNN TTTTTTTTTTTTTTTTT 000000000 RRRRRRRRRRR
MMMMMMM MMMMMMM 000 000 NNN NNN TTT 000 000 RRR RRR
MMMMMMM MMMMMMM 000 000 NNN NNN TTT 000 000 RRR RRR
MMMMMMM MMMMMMM 000 000 NNN NNN TTT 000 000 RRR RRR
MMM MMM MMM 000 000 NNNNNN NNN TTT 000 000 RRR RRR
MMM MMM MMM 000 000 NNNNNN NNN TTT 000 000 RRR RRR
MMM MMM MMM 000 000 NNNNNN NNN TTT 000 000 RRR RRR
MMM MMM 000 000 NNN NNN NNN TTT 000 000 RRRRRRRRRRR
MMM MMM 000 000 NNN NNN NNN TTT 000 000 RRRRRRRRRRR
MMM MMM 000 000 NNN NNN NNN TTT 000 000 RRRRRRRRRRR
MMM MMM 000 000 NNN NNNNN TTT 000 000 RRR RRR
MMM MMM 000 000 NNN NNNNN TTT 000 000 RRR RRR
MMM MMM 000 000 NNN NNNNN TTT 000 000 RRR RRR
MMM MMM 000 000 NNN NNN TTT 000 000 RRR RRR
MMM MMM 000 000 NNN NNN TTT 000 000 RRR RRR
MMM MMM 000 000 NNN NNN TTT 000 000 RRR RRR
MMM MMM 000000000 NNN NNN TTT 000000000 RRR RRR
MMM MMM 000000000 NNN NNN TTT 000000000 RRR RRR
MMM MMM 000000000 NNN NNN TTT 000000000 RRR RRR

```
PPPPPPPP    RRRRRRR    EEEEEEEEEE  PPPPPPPP     000000     SSSSSSSS  TTTTTTTTTT
PPPPPPPP    RRRRRRRR   EEEEEEEEEE  PPPPPPPP     000000     SSSSSSSS  TTTTTTTTTT
PP      PP  RR     RR  EE          PP      PP  00      00  SS            TT
PP      PP  RR      RR EE          PP      PP  00      00  SS            TT
PP      PP  RR      RR EE          PP      PP  00      00  SS            TT
PP      PP  RR     RR  EE          PP      PP  00      00  SS            TT
PPPPPPPP    RRRRRRRR   EEEEEEEE    PPPPPPPP    00      00    SSSSSS      TT
PPPPPPPP    RRRRRRR    EEEEEEEE    PPPPPPPP    00      00    SSSSSS      TT
PP          RR   RR    EE          PP          00      00          SS    TT
PP          RR    RR   EE          PP          00      00          SS    TT
PP          RR     RR  EE          PP          00      00          SS    TT        ....
PP          RR     RR  EE          PP          00      00          SS    TT        ....
PP          RR      RR EEEEEEEEEE  PP            000000    SSSSSSSS      TT        ....
PP          RR      RR EEEEEEEEEE  PP            000000    SSSSSSSS      TT        ....
```

```
LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  PREPOST - VAX/VMS Monitor Pre-post Collection Rtns
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;
0000     5  ;*****************************************************************************
0000     6  ;*                                                                           *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16  ;*  TRANSFERRED.                                                             *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                             *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24  ;*                                                                           *
0000    25  ;*                                                                           *
0000    26  ;*****************************************************************************
0000    27
0000    28  ;++
0000    29  ; FACILITY:  VAX/VMS MONITOR Utility
0000    30  ;
0000    31  ; ABSTRACT:
0000    32  ;
0000    33  ;       The pre- and post- collection routines perform class-specific
0000    34  ;       data collection which does not conform to the scheme required
0000    35  ;       by the FETCH routine.
0000    36  ;
0000    37  ; ENVIRONMENT: Each routine is entered in EXEC mode. Some routines
0000    38  ;                elevate to kernel mode and some additionally raise
0000    39  ;                IPL to synchronize data base access with VMS.
0000    40  ;
0000    41  ; AUTHOR: Henry M. Levy          , CREATION DATE: 28-March-1977
0000    42  ;         Thomas L. Cafarella
0000    43  ;
0000    44  ; MODIFIED BY:
0000    45  ;
0000    46  ;       V03-017 TLC1079         Thomas L. Cafarella     11-Jul-1984     11:00
0000    47  ;               Miscellaneous name and label changes.
0000    48  ;
0000    49  ;       V03-016 TLC1076         Thomas L. Cafarella     09-Jul-1984     15:00
0000    50  ;               Correct reporting of negative queue length for DISK class.
0000    51  ;
0000    52  ;       V03-015 TLC1072         Thomas L. Cafarella     17-Apr-1984     11:00
0000    53  ;               Add volume name to DISK display.
0000    54  ;
0000    55  ;       V03-014 PRS1017         Paul R. Senn            9-Apr-1984      15:00
0000    56  ;               Changes to STATES collection routine to support SYSTEM class
0000    57  ;
```

```
0000     58 ;       V03-013 TLC1056         Thomas L. Cafarella     22-Mar-1984     11:00
0000     59 ;               Disable journaling classes and exclude class which is disabled.
0000     60 ;
0000     61 ;       V03-012 TLC1055         Thomas L. Cafarella     11-Mar-1984     16:00
0000     62 ;               Pick up queue length from UCB for DISK class.
0000     63 ;
0000     64 ;       V03-011 PRS1010         Paul R. Senn            27-Feb-1984      9:00
0000     65 ;               Add precollection routine for DLOCK class
0000     66 ;
0000     67 ;       V03-011 PRS1007         Paul R. Senn            17-FEB-1984     14:00
0000     68 ;               Add precollection routine for XQPCACHE class
0000     69 ;
0000     70 ;       V03-010 PRS1004         Paul R. Senn            11-JAN-1983     16:00
0000     71 ;               Misc. changes to POOL class
0000     72 ;
0000     73 ;       V03-009 SPC0008         Stephen P. Carney       07-Sep-1983     16:00
0000     74 ;               Fix SCS Class Kbyte overflow.
0000     75 ;
0000     76 ;       V03-008 TLC1045         Thomas L. Cafarella     25-Aug-1983     11:00
0000     77 ;               Always include node name in DISK display
0000     78 ;               for cluster systems.
0000     79 ;
0000     80 ;       V03-007 SPC0004         Stephen P. Carney       24-Jun-1983     16:00
0000     81 ;               Add SCS Class pre-collection routine.
0000     82 ;
0000     83 ;       V03-006 TLC1035         Thomas L. Cafarella     06-Jun-1983     15:00
0000     84 ;               Add homogeneous class type and DISK class.
0000     85 ;
0000     86 ;       V03-006 SPC0003         Stephen P. Carney       06-Jun-1983     15:00
0000     87 ;               Add JDEVICE Class pre-collection routine.
0000     88 ;
0000     89 ;       V03-005 TLC1032         Thomas L. Cafarella     27-May-1983     15:00
0000     90 ;               Add Blocking AST Rate to LOCK class.
0000     91 ;
0000     92 ;       V03-004 TLC1028         Thomas L. Cafarella     14-Apr-1983     16:00
0000     93 ;               Add interactive user interface.
0000     94 ;
0000     95 ;       V03-004 TLC1027         Thomas L. Cafarella     14-Apr-1983     16:00
0000     96 ;               Enhance file compatibility features.
0000     97 ;
0000     98 ;       V03-004 TLC1026         Thomas L. Cafarella     14-Apr-1983     16:00
0000     99 ;               Miscellaneous updates to JOURNALING, RU and FCP classes.
0000    100 ;
0000    101 ;       V03-003 KDM0002         Kathleen D. Morse       28-Jun-1982
0000    102 ;               Added $PRDEF.
0000    103 ;
0000    104 ;--
```

```
                        0000    106
                        0000    107             .SBTTL  DECLARATIONS
                    00000000    108             .PSECT  DSPDATA,QUAD,NOEXE
                        0000    109 ;
                        0000    110 ; INCLUDE FILES:
                        0000    111 ;
                        0000    112
                        0000    113             $CDTDEF                         ; Define Connection Desc. Table offsets
                        0000    114             $DCDEF                          ; Define device class codes
                        0000    115             $DEVDEF                         ; Define device characteristics flags
                        0000    116             $DDBDEF                         ; Define Device Data Block offsets
                        0000    117             $IPLDEF                         ; Define Interrupt Processor Levels
                        0000    118             $IRPDEF                         ; Define Intermediate req. pkt. offsets
                        0000    119             $PBDEF                          ; Define Path Block offsets
                        0000    120             $PCBDEF                         ; Process control block definitions
                        0000    121             $PHDDEF                         ; Process header definitions
                        0000    122             $STATEDEF                       ; Process state definitions
                        0000    123             $PRDEF                          ; Define processor register numbers
                        0000    124             $SBDEF                          ; Define System Block offsets
                        0000    125             $UCBDEF                         ; Define Unit Control Block offsets
                        0000    126             $VCBDEF                         ; Define Volume Control Block offsets
                        0000    127             $CDBDEF                         ; Define Class Descriptor Block
                        0000    128             $MRBDEF                         ; Define Monitor Request Block
                        0000    129             $MBPDEF                         ; Define Monitor Buffer Pointers
                        0000    130             $MCADEF                         ; Define Monitor Communication Area
                        0000    131             $MONDEF                         ; Monitor Recording File Definitions
                        0000    132
                        0000    133 ;
                        0000    134 ; MACROS:
                        0000    135 ;
                        0000    136
                        0000    137 ;
                        0000    138 ; Local Macro Definitions
                        0000    139 ;
                        0000    140
                        0000    141 ;
                        0000    142 ; ALLOC Macro - Dynamically allocate space on the stack.
                        0000    143 ;
                        0000    144
                        0000    145             .MACRO  ALLOC   LENGTH,RSLDESC,RSLBUF
                        0000    146             SUBL    #<LENGTH+3>&<^C3>,SP
                        0000    147             .IF     NB,RSLBUF
                        0000    148             MOVL    SP,RSLBUF
                        0000    149             .ENDC
                        0000    150             PUSHL   SP
                        0000    151             PUSHL   #LENGTH
                        0000    152             MOVL    SP,RSLDESC
                        0000    153             .ENDM   ALLOC
                        0000    154
```

```
                     0000   156 ;
                     0000   157 ; EQUATED SYMBOLS:
                     0000   158 ;
                     0000   159
                     0000   160 ;
                     0000   161 ; SCS class symbols for collection buffer offset.
                     0000   162 ;
                     0000   163
00000000             0000   164 MNR_SCS$Q_NODENAME      = 00          ; SCS counted ASCII node name
00000008             0000   165 MNR_SCS$L_DGSENT        = 08          ; SCS application datagrams sent
0000000C             0000   166 MNR_SCS$L_DGRCVD        = 12          ; SCS application datagrams received
00000010             0000   167 MNR_SCS$L_DGDISCARD     = 16          ; SCS application datagrams discarded
00000014             0000   168 MNR_SCS$L_MSGSENT       = 20          ; SCS application messages sent
00000018             0000   169 MNR_SCS$L_MSGRCVD       = 24          ; SCS application messages received
0000001C             0000   170 MNR_SCS$L_SNDDATS       = 28          ; SCS block send datas initiated
00000020             0000   171 MNR_SCS$L_KBYTSENT      = 32          ; SCS Kbytes sent via send datas
00000024             0000   172 MNR_SCS$L_REQDATS       = 36          ; SCS block request datas initiated
00000028             0000   173 MNR_SCS$L_KBYTREQD      = 40          ; SCS Kbytes received via request datas
0000002C             0000   174 MNR_SCS$L_KBYTMAPD      = 44          ; SCS Kbytes mapped for block xfr
00000030             0000   175 MNR_SCS$L_QCR_CNT       = 48          ; SCS times conn. q'd for send credit
00000034             0000   176 MNR_SCS$L_QBDT_CNT      = 52          ; SCS times conn. q'd for buff descr
                     0000   177
00000038             0000   178 MNR_SCS$L_CBKBSENT      = 56          ; SCS aux coll. buff. to cvt KB sent
0000003C             0000   179 MNR_SCS$L_CBKBREQD      = 60          ; SCS aux coll. buff. to cvt KB request
00000040             0000   180 MNR_SCS$L_CBKBMAPD      = 64          ; SCS aux coll. buff. to cvt KB map
                     0000   181
00000038             0000   182 MNR_SCS$C_CBLENGTH      = 56          ; Length of one collection
00000044             0000   183 MNR_SCS$C_CBWORK        = 68          ; Extra working space in coll. buff.
                     0000   184
                     0000   185
```

```
                                E 2
                     0000   187 ;
                     0000   188 ; OWN STORAGE:
                     0000   189 ;
                     0000   190
                     0000   191
00000004             0000   192 FCPCALLS::      .BLKL   1               ; total calls to FCP
00000008             0004   193 FCPCACHE::      .BLKL   1               ; FCP directory cache hits
0000000C             0008   194 FCPCPU::        .BLKL   1               ; FCP CPU time used
00000010             000C   195 FCPREAD::       .BLKL   1               ; FCP disk reads
00000014             0010   196 FCPWRITE::      .BLKL   1               ; FCP disk writes
00000018             0014   197 FCPFAULT::      .BLKL   1               ; FCP page faults
                     0018   198
                     0018   199 ;
                     0018   200 ; Space for accumulating statistics on the nonpaged pool.
                     0018   201 ; (do not change order)
                     0018   202 ;
                     0018   203
0000001C             0018   204 HOLECNT::       .BLKL   1               ; number of blocks in nonpaged pool
00000020             001C   205 HOLESUM::       .BLKL   1               ; total space in pool
00000024             0020   206 BIGHOLE::       .BLKL   1               ; largest hole in pool
00000028             0024   207 SMALLCNT::      .BLKL   1               ; number of holes < 32 bytes
0000002C             0028   208 SMALLHOLE::     .BLKL   1               ; smallest hole in pool
00000030             002C   209 IRPCNT::        .BLKL   1               ; number of I/O (intermed) request packets
00000034             0030   210 LRPCNT::        .BLKL   1               ; number of large request packets
00000038             0034   211 SRPCNT::        .BLKL   1               ; number of small request packets
0000003C             0038   212 SRPINUSE::      .BLKL   1               ; number of SRPs in use
00000040             003C   213 IRPINUSE::      .BLKL   1               ; number of IRPs in use
00000044             0040   214 LRPINUSE::      .BLKL   1               ; number of LRPs in use
00000048             0044   215 DYNINUSE::      .BLKL   1               ; size in bytes of variable part
                     0048   216                                         ; of nonpaged pool currently in use
0000004C             0048   217 SYSFAULTS::     .BLKL   1               ; count of system space page faults
                     004C   218
                     004C   219 ;
                     004C   220 ; Data for the Lock class
                     004C   221 ;
                     004C   222
00000050             004C   223 ENQNEW::        .BLKL   1               ; new ENQs
00000054             0050   224 ENQCVT::        .BLKL   1               ; converted ENQs
00000058             0054   225 DEQ::           .BLKL   1               ; DEQs
0000005C             0058   226 BLKAST::        .BLKL   1               ; blocking ASTs
                     005C   227
00000060             005C   228 LOCKCNT::       .BLKL   1               ; current number of locks in the system
00000064             0060   229 RESCNT::        .BLKL   1               ; current number of resources in the system
                     0064   230
                     0064   231 ;
                     0064   232 ; Data for the DLock class
                     0064   233 ;
                     0064   234
00000068             0064   235 DLCKMSGS::      .BLKL   1               ; Messages send to do Deadlock detection
                     0068   236
                     0068   237 ;
                     0068   238 ; Data for the MODES class
                     0068   239 ;
                     0068   240
0000006C             0068   241 CPU_BUSY::      .BLKL   1               ; sum of the 6 mode counters
00000074             006C   242 MPSTRTIM:       .BLKQ   1               ; save area for MP start time
00000000 00000000 00000000 00000000  0074   243 BASE:  .LONG 0,0,0,0,0,0,0     ; 7 Secondary base counter values
```

```
00000000 00000000 00000000  0084
                            0090  244
                            0090  245 ;
                            0090  246 ; Data for the STATES class (used by SYSTEM class)
                            0090  247 ;
                   00000094 0090  248 PROC_COUNT::     .BLKL   1               ; Sum of all processes
                   00000098 0094  249 OTHER_STATES::   .BLKL   1               ; Sum of processes in OTHER category
                            0098  250                                          ; on system manager's screen.
                            0098  251 SYSMGR_STATES:                           ; array of states shown on
                            0098  252                                          ; SYSTEM screen (all others are OTHER)
                         02 0098  253                   .BYTE   SCH$C_MWAIT
                         04 0099  254                   .BYTE   SCH$C_PFW
                         05 009A  255                   .BYTE   SCH$C_LEF
                         06 009B  256                   .BYTE   SCH$C_LEFO
                         07 009C  257                   .BYTE   SCH$C_HIB
                         08 009D  258                   .BYTE   SCH$C_HIBO
                         0C 009E  259                   .BYTE   SCH$C_COM
                         0D 009F  260                   .BYTE   SCH$C_COMO
                            00A0  261
                   00000008 00A0  262 SYSMGR_STATETOT = <. - SYSMGR_STATES>    ; Number of states on SYSTEM screen
                            00A0  263
                            00A0  264 ;
                            00A0  265 ; Data for the FILE_SYSTEM_CACHE class
                            00A0  266 ;
                            00A0  267
                   000000A4 00A0  268 FILHDR_TRIES::   .BLKL   1               ; hits + misses on File Header cache
                   000000A8 00A4  269 FID_TRIES::      .BLKL   1               ; hits + misses on FID cache
                   000000AC 00A8  270 DIRFCB_TRIES::   .BLKL   1               ; hits + misses on Directory FCB cache
                   000000B0 00AC  271 DIRDATA_TRIES::  .BLKL   1               ; hits + misses on Directory Data cache
                   000000B4 00B0  272 EXT_TRIES::      .BLKL   1               ; hits + misses on Extent cache
                   000000B8 00B4  273 QUO_TRIES::      .BLKL   1               ; hits + misses on Quota cache
                            00B8  274 STORAGMAP_TRIES::
                   000000BC 00B8  275                   .BLKL   1               ; hits + misses on Storage bitmap cache
                            00BC  276
                            00BC  277 ;
                            00BC  278 ; Data for the DISK class
                            00BC  279 ;
                            00BC  280
                20 20 20 20 00BC  281 BLANKS:          .ASCII  \    \          ; used to collect a non-existent volnam
```

```
                                                G 2
                     00C0   283          .SBTTL  FCP_PRE - FCP Class Pre-collection Rtn
                00000000    284          .PSECT  $$MONCODE,NOWRT,EXE
                     0000   285 ;++
                     0000   286 ;
                     0000   287 ; FUNCTIONAL DESCRIPTION:
                     0000   288 ;
                     0000   289 ;       This routine accumulates statistics from the File Control Primitive
                     0000   290 ;       data base and saves them in global variables so that they
                     0000   291 ;       may be fetched and processed by the standard FETCH
                     0000   292 ;       collection routine.
                     0000   293 ;
                     0000   294 ; CALLING SEQUENCE:
                     0000   295 ;
                     0000   296 ;       CALLS/CALLG
                     0000   297 ;
                     0000   298 ; INPUTS:
                     0000   299 ;
                     0000   300 ;       4(AP) - address of current collection buffer (unused by this rtn)
                     0000   301 ;
                     0000   302 ; IMPLICIT INPUTS:
                     0000   303 ;
                     0000   304 ;       PMS$GL_FCP2 - pointer to ten arrays of FCP data
                     0000   305 ;
                     0000   306 ; OUTPUTS:
                     0000   307 ;
                     0000   308 ;       None
                     0000   309 ;
                     0000   310 ; IMPLICIT OUTPUTS:
                     0000   311 ;
                     0000   312 ;       FCPCALLS - contains total calls made to FCP
                     0000   313 ;       FCPCACHE - total FCP cache hits
                     0000   314 ;       FCPCPU - percent of CPU time used by FCP during the last
                     0000   315 ;                interval
                     0000   316 ;       FCPREAD - total FCP disk reads
                     0000   317 ;       FCPWRITE - total FCP disk writes
                     0000   318 ;       FCPFAULT - total FCP page faults
                     0000   319 ;
                     0000   320 ; ROUTINE VALUE:
                     0000   321 ;
                     0000   322 ;       R0 = SS$_NORMAL
                     0000   323 ;
                     0000   324 ;       R1 = YES, if subsequent FETCH collection is required.
                     0000   325 ;       R1 = NO,  if subsequent FETCH collection is NOT required.
                     0000   326 ;
                     0000   327 ; SIDE EFFECTS:
                     0000   328 ;
                     0000   329 ;       none
                     0000   330 ;--
                     0000   331
               0000  0000   332 .ENTRY  FCP_PRE, ^M<>
                     0002   333
                     0002   334 ;
                     0002   335 ; Compute total calls to fcp
                     0002   336 ;
                     0002   337
     50    05    D0  0002   338          MOVL    #5,R0                   ; sum first six counters
0000'CF    D4    0005   339          CLRL    W^FCPCALLS              ; clear counter
```

```
                        0009   340 10$:
0000'CF  00000000'EF40  C0 0009   341         ADDL    PMS$GL_FCP2[R0],W^FCPCALLS ; add in next counter
             F3 50      F4 0013   342         SOBGEQ  R0,10$                     ; continue till done
                        0016   343
                        0016   344 ;
                        0016   345 ; Compute disk reads and writes, cache hits, % CPU TIME and faults
                        0016   346 ;
                        0016   347
          50    09      D0 0016   348         MOVL    #9,R0                      ; sum 10 entries in each array
          000C'CF       7C 0019   349         CLRQ    W^FCPREAD                  ; clear reads and writes
          0004'CF       7C 001D   350         CLRQ    W^FCPCACHE                 ; clear cache and cpu time
          0014'CF       D4 0021   351         CLRL    W^FCPFAULT                 ; clear page faults
                        0025   352 20$:
000C'CF  00000050'EF40  C0 0025   353         ADDL    PMS$GL_FCP2+<20*4>[R0],W^FCPREAD ; sum reads
0010'CF  00000078'EF40  C0 002F   354         ADDL    PMS$GL_FCP2+<30*4>[R0],W^FCPWRITE ; sum writes
0004'CF  000000A0'EF40  C0 0039   355         ADDL    PMS$GL_FCP2+<40*4>[R0],W^FCPCACHE ; cache hits
0008'CF  000000C8'EF40  C0 0043   356         ADDL    PMS$GL_FCP2+<50*4>[R0],W^FCPCPU ; sum cpu tics used
0014'CF  000000F0'EF40  C0 004D   357         ADDL    PMS$GL_FCP2+<60*4>[R0],W^FCPFAULT ; sum page faults
             CB 50      F4 0057   358         SOBGEQ  R0,20$
                        005A   359 ;
                        005A   360 ; Indicate to caller that FETCH collection IS required.
                        005A   361 ;
                        005A   362
      51  00000000'8F   D0 005A   363         MOVL    #YES,R1                    ; FETCH collection required
      50  00000000'8F   D0 0061   364         MOVL    #SS$_NORMAL,R0             ; success status
                        04 0068   365         RET                                ; return
```

```
                                                     .SBTTL  POOL_PRE - Pre-collection for Pool Statistics

                                                ;++
                                                ; FUNCTIONAL DESCRIPTION:
                                                ;
                                                ;       Routine to accumulate statistics on behavior of SRP/IRP/LRP
                                                ;       lookaside lists and nonpaged dynamic memory pool.
                                                ;
                                                ; CALLING SEQUENCE:
                                                ;
                                                ;       CALLS/CALLG
                                                ;
                                                ; INPUTS:
                                                ;
                                                ;       4(AP) - address of current collection buffer (unused by this rtn).
                                                ;
                                                ; IMPLICIT INPUTS:
                                                ;
                                                ;       none
                                                ;
                                                ; OUTPUTS:
                                                ;
                                                ;       none
                                                ;
                                                ; IMPLICIT OUTPUTS:
                                                ;
                                                ;       LRPCNT, IRPCNT, SRPCNT, HOLECNT, BIGHOLE, SMALLHOLE,
                                                ;       SMALLCNT, SRPINUSE, IRPINUSE, LRPINUSE, DYNINUSE and HOLESUM
                                                ;       are set by subroutine SCANPOOL
                                                ;
                                                ; ROUTINE VALUE:
                                                ;
                                                ;       R0 = SS$_NORMAL
                                                ;
                                                ;       R1 = YES, if subsequent FETCH collection is required.
                                                ;       R1 = NO,  if subsequent FETCH collection is NOT required.
                                                ;
                                                ; SIDE EFFECTS:
                                                ;
                                                ;       none
                                                ;--
                                                
                                        0000    .ENTRY  POOL_PRE, ^M<>
                                                        $CMKRNL_S B^SCANPOOL            ; get stats in kernel mode
51  00000000'8F    D0                           MOVL    #YES,R1                         ; indicate FETCH collection IS required
50  00000000'8F    D0                           MOVL    #SS$_NORMAL,R0                  ; success status
                   04                           RET                                     ; return
```

```
                     0086   417 ;++
                     0086   418 ; SCANPOOL - subroutine to update pool statistics
                     0086   419 ;
                     0086   420 ; CALLING SEQUENCE:
                     0086   421 ;
                     0086   422 ;       $CMKRNL_S SCANPOOL
                     0086   423 ;
                     0086   424 ; IMPLICIT INPUTS:
                     0086   425 ;
                     0086   426 ;       IOC$GL_SRPFL - address of SRP listhead
                     0086   427 ;       IOC$GL_IRPFL - address of IRP listhead
                     0086   428 ;       IOC$GL_LRPFL - address of LRP listhead
                     0086   429 ;       IOC$GL_SRPCNT - total number of SRP packets (used + available)
                     0086   430 ;       IOC$GL_IRPCNT - total number of IRP packets (used + available)
                     0086   431 ;       IOC$GL_LRPCNT - total number of LRP packets (used + available)
                     0086   432 ;       EXE$GL_NONPAGED - address of nonpaged pool listhead
                     0086   433 ;
                     0086   434 ; IMPLICIT OUTPUTS:
                     0086   435 ;
                     0086   436 ;       SRPCNT - number of SRP packets available
                     0086   437 ;       IRPCNT - number of IRP packets available
                     0086   438 ;       LRPCNT - number of LRP packets available
                     0086   439 ;       SRPINUSE - Number of SRP packets in use
                     0086   440 ;       IRPINUSE - Number of IRP packets in use
                     0086   441 ;       LRPINUSE - Number of LRP packets in use
                     0086   442 ;       DYNINUSE - Size of variable nonpaged pool in use (in bytes)
                     0086   443 ;       HOLECNT - number of memory blocks in NONPAGED pool
                     0086   444 ;       BIGHOLE - largest memory block
                     0086   445 ;       SMALLHOLE - smallest memory block
                     0086   446 ;       SMALLCNT - number of 32 byte or smaller blocks
                     0086   447 ;       HOLESUM - total space in nonpaged pool
                     0086   448 ;
                     0086   449 ; SIDE EFFECTS:
                     0086   450 ;
                     0086   451 ;       must synchronize data base
                     0086   452 ;--
                     0086   453
                     0086   454 SCANPOOL:
             0FFC    0086   455         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; register save mask
                     0088   456
                     0088   457 ;
                     0088   458 ; Initialize all variables possible at this level.
                     0088   459 ;
                     0088   460
       52    7C      0088   461         CLRQ    R2                      ; clear holecnt, holesum
       54    7C      008A   462         CLRQ    R4                      ; clear for bighole, smallcnt
  56   01    CE      008C   463         MNEGL   #1,R6                   ; make smallest hole very large
       57    D4      008F   464         CLRL    R7                      ; clear for IRP counter
       59    7C      0091   465         CLRQ    R9                      ; clear for LRP, SRP counters
                     0093   466
                     0093   467 ;
                     0093   468 ; Touch last word of sequence to make sure all code is resident.
                     0093   469 ;
                     0093   470
0139'CF      D5      0093   471         TSTL    W^120$                  ; make sure all code is resident
                     0097   472
                     0097   473 ;
```

```
                                0097   474 ; Save address of nonpaged listhead and run at IPL
                                0097   475 ; contained there.
                                0097   476 ;
                                0097   477
            58  00000000'EF    DE  0097   478         MOVAL   EXE$GL_NONPAGED,R8      ; get nonpaged pool listhead
                                009E   479 5$:      DSBINT  (R8)+,#R11              ; set ipl for pool access
                                00A4   480
                                00A4   481 ;
                                00A4   482 ; Get the current total # of packets of each type and save on the stack
                                00A4   483 ;
                                00A4   484
                00000000'EF     DD  00A4   485         PUSHL   IOC$GL_SRPCNT           ; Save total SRPs
                00000000'EF     DD  00AA   486         PUSHL   IOC$GL_IRPCNT           ; Save total IRPs
                00000000'EF     DD  00B0   487         PUSHL   IOC$GL_LRPCNT           ; Save total LRPs
                                00B6   488
                                00B6   489 ;
                                00B6   490 ; Get the current total size of variable pool in bytes and save on stack
                                00B6   491 ;
50  00000000'GF  000001FF 8F    CB  00B6   492         BICL3   #^X1FF,G^MMG$GL_NPAGNEXT,R0     ; Get current end of pool
      7E  50    00000000'GF    C3  00C2   493         SUBL3   G^MMG$GL_NPAGEDVN,R0,-(SP)      ; Compute pool size
                                00CA   494                                                 ; and save on the stack
                                00CA   495 ;
                                00CA   496 ; Run through the SRP list and count the packets remaining
                                00CA   497 ;
                                00CA   498
            50  00000000'EF     DE  00CA   499         MOVAL   IOC$GL_SRPFL,R0         ; get SRP listhead address
                  51    50      D0  00D1   500         MOVL    R0,R1                   ; copy header address
                                00D4   501
                  51    61      D0  00D4   502 10$:     MOVL    (R1),R1                 ; get forward link
                  50    51      D1  00D7   503         CMPL    R1,R0                   ; point back to header?
                        04      13  00DA   504         BEQL    20$                     ; done if so
                        5A      D6  00DC   505         INCL    R10                     ; count one more packet
                        F4      11  00DE   506         BRB     10$                     ; loop back for more
                                00E0   507 20$:
                                00E0   508
                                00E0   509 ;
                                00E0   510 ; Run through the IRP list and count the packets remaining
                                00E0   511 ;
                                00E0   512
50  00000000'EF     DE  00E0   513         MOVAL   IOC$GL_IRPFL,R0         ; get IRP listhead address
      51    50      D0  00E7   514         MOVL    R0,R1                   ; copy header address
                                00EA   515
            51    61      D0  00EA   516 30$:     MOVL    (R1),R1                 ; get forward link
            50    51      D1  00ED   517         CMPL    R1,R0                   ; point back to header?
                  04      13  00F0   518         BEQL    40$                     ; done if so
                  57      D6  00F2   519         INCL    R7                      ; count one more packet
                  F4      11  00F4   520         BRB     30$                     ; loop back for more
                                00F6   521 40$:
                                00F6   522
                                00F6   523 ;
                                00F6   524 ; Run through the LRP list and count the packets remaining
                                00F6   525 ;
                                00F6   526
50  00000000'EF     DE  00F6   527         MOVAL   IOC$GL_LRPFL,R0         ; get LRP listhead address
      51    50      D0  00FD   528         MOVL    R0,R1                   ; copy header address
                                0100   529
            51    61      D0  0100   530 50$:     MOVL    (R1),R1                 ; get forward link
```

```
                                                                L 2
                     50    51    D1  0103    531          CMPL    R1,R0                   ; point back to header?
                           04    13  0106    532          BEQL    60$                     ; done if so
                           59    D6  0108    533          INCL    R9                      ; count one more packet
                           F4    11  010A    534          BRB     50$                     ; loop back for more
                                     010C    535 60$:
                                     010C    536
                                     010C    537 ;
                                     010C    538 ; Now run through the nonpaged pool, count the blocks, and check the
                                     010C    539 ; smallest and largest holes.
                                     010C    540 ;
                                     010C    541
                     50    58    D0  010C    542          MOVL    R8,R0                   ; get pool listhead address
                     50    60    D0  010F    543 70$:     MOVL    (R0),R0                 ; get address of next block
                           22    13  0112    544          BEQL    110$                    ; branch if zero, list done
                           52    D6  0114    545          INCL    R2                      ; note one more block
                   51    04    A0  D0  0116    546          MOVL    4(R0),R1                ; get size of block
                     53    51    C0  011A    547          ADDL    R1,R3                   ; add in size of this block
                     56    51    D1  011D    548          CMPL    R1,R6                   ; is this smallest found?
                           03    1E  0120    549          BGEQU   80$                     ; branch if not
                     56    51    D0  0122    550          MOVL    R1,R6                   ; else save it
                     54    51    D1  0125    551 80$:     CMPL    R1,R4                   ; is this largest found?
                           03    1B  0128    552          BLEQU   90$                     ; branch if not
                     54    51    D0  012A    553          MOVL    R1,R4                   ; else update largest
                     20    51    D1  012D    554 90$:     CMPL    R1,#32                  ; is this one of the small ones?
                           02    1A  0130    555          BGTRU   100$                    ; branch if not
                           55    D6  0132    556          INCL    R5                      ; note another small hole
                           D9    11  0134    557 100$:    BRB     70$                     ; go on to next block
                                     0136    558 110$:    ENBINT  R11                     ; enable interrupts
                                     0139    559                                         ; of non-paged pool in use
                                     0139    560          ASSUME .-5$ LE 512             ; must be on one page only
          0018'CF    52    7D  0139    561 120$:    MOVQ    R2,W^HOLECNT            ; save variables (HOLECNT and HOLESUM)
          0020'CF    54    7D  013E    562           MOVQ    R4,W^BIGHOLE            ; ...
          0028'CF    56    7D  0143    563           MOVQ    R6,W^SMALLHOLE          ; ...
          0030'CF    59    7D  0148    564           MOVQ    R9,W^LRPCNT             ; ...
00000044'EF    8E    53    C3  014D    565           SUBL3   R3,(SP)+,DYNINUSE       ; Calculate and save dynamic mem in use
00000040'EF    8E    59    C3  0155    566           SUBL3   R9,(SP)+,LRPINUSE       ; Calculate and save LRPs in use
0000003C'EF    8E    57    C3  015D    567           SUBL3   R7,(SP)+,IRPINUSE       ; Calculate and save IRPs in use
00000038'EF    8E    5A    C3  0165    568           SUBL3   R10,(SP)+,SRPINUSE      ; Calculate and save SRPs in use
                           04  016D    569           RET
```

```
                                                M 2
                                                .SBTTL  LOCK_PRE - Pre-collection for Lock Statistics

                                ;++
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       Routine to count the number of locks and resources in the system,
                                ;       and to total LOCK counters for incoming, outgoing, and local.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       CALLS/CALLG
                                ;
                                ; INPUTS:
                                ;
                                ;       None
                                ;
                                ; IMPLICIT INPUTS:
                                ;
                                ;       LCK$GL_IDTBL    Contains address of lock id table
                                ;       LCK$GL_MAXID    Contains maximum lock id
                                ;       LCK$GL_HASHTBL  Contains address of resource hash table
                                ;       LCK$GL_HTBLCNT  Contains # entries in hash table (expresses as a
                                ;                       power of two)
                                ;
                                ; OUTPUTS:
                                ;
                                ;       None
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;
                                ;       ENQNEW, ENQCVT, DEQ, BLKAST, LOCKCNT and RESCNT are set.
                                ;
                                ; ROUTINE VALUE:
                                ;
                                ;       R0 = SS$_NORMAL
                                ;
                                ;       R1 = YES, if subsequent FETCH collection is required.
                                ;       R1 = NO,  if subsequent FETCH collection is NOT required.
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       None
                                ;--
                                
                        003C    .ENTRY  LOCK_PRE, ^M<R2,R3,R4,R5>
                                ;
                                ; Initialize to count the number of locks
                                ;
                                
55  00000000'GF   D0            MOVL    G^LCK$GL_IDTBL,R5       ; Get address of lock id table
54  00000000'GF   D0            MOVL    G^LCK$GL_MAXID,R4       ; Get maximum lock id
              53  D4            CLRL    R3                      ; Initialize counter of locks
                                ;
                                ; Count the number of locks
```

```
                                        0180   628 ;
                                        0180   629
                              85   D5   0180   630 10$:     TSTL    (R5)+                   ; Is there a lock in this slot?
                              02   18   0182   631         BGEQ    20$                     ; No
                              53   D6   0184   632         INCL    R3                      ; Yes, bump counter
                           F7 54   F4   0186   633 20$:     SOBGEQ  R4,10$                  ; Repeat for all entries in table
                    0000005C'EF   53   D0   0189   634         MOVL    R3,LOCKCNT              ; Store final value
                                        0190   635
                                        0190   636 ;
                                        0190   637 ; Count the number of resources
                                        0190   638 ;
                                        0190   639
                                        0190   640         $CMKRNL_S       B^COUNT_RES     ; Do it in kernel mode
                                        019C   641
                                        019C   642 ;
                                        019C   643 ; Total local, incoming and outgoing counters for
                                        019C   644 ; ENQNEW, ENQCVT, DEQ and BLKAST.
                                        019C   645 ;
                                        019C   646
52   00000000'EF   00000000'EF   C1     019C   647         ADDL3   PMS$GL_ENQNEW_LOC,PMS$GL_ENQNEW_IN,R2
     004C'CF   52   00000000'EF   C1     01A8   648         ADDL3   PMS$GL_ENQNEW_OUT,R2,W^ENQNEW
                                        01B2   649
52   00000000'EF   00000000'EF   C1     01B2   650         ADDL3   PMS$GL_ENQCVT_LOC,PMS$GL_ENQCVT_IN,R2
     0050'CF   52   00000000'EF   C1     01BE   651         ADDL3   PMS$GL_ENQCVT_OUT,R2,W^ENQCVT
                                        01C8   652
52   00000000'EF   00000000'EF   C1     01C8   653         ADDL3   PMS$GL_DEQ_LOC,PMS$GL_DEQ_IN,R2
     0054'CF   52   00000000'EF   C1     01D4   654         ADDL3   PMS$GL_DEQ_OUT,R2,W^DEQ
                                        01DE   655
52   00000000'EF   00000000'EF   C1     01DE   656         ADDL3   PMS$GL_BLK_LOC,PMS$GL_BLK_IN,R2
     0058'CF   52   00000000'EF   C1     01EA   657         ADDL3   PMS$GL_BLK_OUT,R2,W^BLKAST
                                        01F4   658
              51   00000000'8F   D0     01F4   659         MOVL    #YES,R1                 ; Indicate FETCH collection IS required
              50   00000000'8F   D0     01FB   660         MOVL    #SS$_NORMAL,R0          ; Success status
                                   04   0202   661         RET
```

```
                                      0203  ;++
                                      0203  ; COUNT_RES - Routine to count resources
                                      0203  ;
                                      0203  ; CALLING SEQUENCE:
                                      0203  ;
                                      0203  ;       $CMKRNL_S       COUNT_RES
                                      0203  ;
                                      0203  ; IMPLICIT INPUTS:
                                      0203  ;
                                      0203  ;       LCK$GL_HASHTBL  Contains address of resource hash table
                                      0203  ;       LCK$GL_HTBLCNT  Contains # entries in hash table (expresses as a
                                      0203  ;                       power of two)
                                      0203  ;
                                      0203  ; IMPLICIT OUTPUTS:
                                      0203  ;
                                      0203  ;       RESCNT - Number of resources
                                      0203  ;
                                      0203  ; SIDE EFFECTS:
                                      0203  ;
                                      0203  ;       Must raise IPL to synchronize database access
                                      0203  ;--
                                      0203
                                      0203  COUNT_RES:
                                003C  0203          .WORD   ^M<R2,R3,R4,R5>
                                      0205
                                      0205  ;
                                      0205  ; Initialize to count resources
                                      0205  ;
                                      0205
55   00000000'GF        D0  0205          MOVL    G^LCK$GL_HASHTBL,R5     ; Get address of hash table
50   00000000'GF        D0  020C          MOVL    G^LCK$GL_HTBLCNT,R0     ; Get size of table as power of two
   54    01    50        78  0213          ASHL    R0,#1,R4                ; Convert to number of entries
               53        D4  0217          CLRL    R3                      ; Initialize resource counter
                                      0219
                                      0219  ;
                                      0219  ; Count resources
                                      0219  ;
                                      0219
          50    85        DE  0219  20$:    MOVAL   (R5)+,R0                ; Get address of next list head
                              021C          SETIPL  50$                     ; Raise IPL (and lock pages in w.s.)
          50    60        D0  0223  30$:    MOVL    (R0),R0                 ; Get next element in list
                04        13  0226          BEQL    40$                     ; Reached end of list
                53        D6  0228          INCL    R3                      ; Bump counter
                F7        11  022A          BRB     30$                     ; Continue down list
                              022C  40$:    SETIPL  #0                      ; Lower IPL
             E7 54        F5  022F          SOBGTR  R4,20$                  ; Repeat for next list
    00000060'EF 53        D0  0232          MOVL    R3,RESCNT               ; Store final value
                          04  0239          RET
                              023A
                    00000008  023A  50$:    .LONG   IPL$_SYNCH
                              023E          ASSUME  .-20$  LE  512          ; Make sure it doesn't exceed two pages
                              023E
```

```
                                        .SBTTL  DLOCK_PRE - Pre-collection for Distributed Lock Statistics

                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       Routine to get the number of SCS messages sent in the service
                                ;       of deadlock detection.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       CALLS/CALLG
                                ;
                                ; INPUTS:
                                ;
                                ;       None
                                ;
                                ; IMPLICIT INPUTS:
                                ;
                                ;       PMS$GL_DLCKMSGS_IN - Deadlock detection messages recieved
                                ;       PMS$GL_DLCKMSGS_OUT - Deadlock detection messages sent
                                ;
                                ; OUTPUTS:
                                ;
                                ;       None
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;
                                ;       DLCKMSGS is set.
                                ;
                                ; ROUTINE VALUE:
                                ;
                                ;       R0 = SS$_NORMAL
                                ;
                                ;       R1 = YES, if subsequent FETCH collection is required.
                                ;       R1 = NO,  if subsequent FETCH collection is NOT required.
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       None
                                ;--
                                
                        0000    .ENTRY  DLOCK_PRE, ^M<>
                                
00000000'EF 00000000'EF C1      ADDL3   PMS$GL_DLCKMSGS_IN, -
            00000064'EF                 PMS$GL_DLCKMSGS_OUT, DLCKMSGS
          51 00000000'8F D0     MOVL    #YES,R1                 ; Indicate FETCH collection IS required
          50 00000000'8F D0     MOVL    #SS$_NORMAL,R0          ; Success status
                        04      RET
```

```
                                        .SBTTL  DECNET_PRE - Pre-collection for DECnet Statistics

                                ;++
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       Routine to calculate current size of LRP lookaside
                                ;       list for inclusion in the DECNET class.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       CALLS/CALLG
                                ;
                                ; INPUTS:
                                ;
                                ;       4(AP) - address of current collection buffer (unused by this rtn).
                                ;
                                ; IMPLICIT INPUTS:
                                ;
                                ;       none
                                ;
                                ; OUTPUTS:
                                ;
                                ;       none
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;
                                ;       LRPCNT is set by subroutine SCANLRP.
                                ;
                                ; ROUTINE VALUE:
                                ;
                                ;       R0 = SS$_NORMAL
                                ;
                                ;       R1 = YES, if subsequent FETCH collection is required.
                                ;       R1 = NO,  if subsequent FETCH collection is NOT required.
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       none
                                ;--
                                ;
                        0000    .ENTRY  DECNET_PRE, ^M<>
                                ;
                                        $CMKRNL_S B^SCANLRP             ; scan LRP list in kernel mode
51  00000000'8F    D0           MOVL    #YES,R1                         ; indicate FETCH collection IS required
50  00000000'8F    D0           MOVL    #SS$_NORMAL,R0                  ; success status
                   04           RET                                     ; return
```

```
                                                E 3
                     027C   814 ;++
                     027C   815 ; SCANLRP - subroutine to calculate LRP count
                     027C   816 ;
                     027C   817 ; CALLING SEQUENCE:
                     027C   818 ;
                     027C   819 ;       $CMKRNL_S SCANLRP
                     027C   820 ;
                     027C   821 ; IMPLICIT INPUTS:
                     027C   822 ;
                     027C   823 ;       IOC$GL_LRPFL - address of LRP listhead
                     027C   824 ;       EXE$GL_NONPAGED - address of nonpaged pool listhead
                     027C   825 ;
                     027C   826 ; IMPLICIT OUTPUTS:
                     027C   827 ;
                     027C   828 ;       LRPCNT - number of packets in LRP list
                     027C   829 ;
                     027C   830 ; SIDE EFFECTS:
                     027C   831 ;
                     027C   832 ;       must synchronize data base
                     027C   833 ;--
                     027C   834
                     027C   835 SCANLRP:
                0000 027C   836         .WORD   ^M<R2,R3>               ; register save mask
                     027E   837
                     027E   838
                  53 D4 027E   839         CLRL    R3                      ; clear LRP counter
                     0280   840
                     0280   841 ;
                     0280   842 ; Touch last word of sequence to make sure all code is resident.
                     0280   843 ;
                     0280   844
                A9'AF D5 0280   845         TSTL    B^30$                   ; make sure all code is resident
                     0283   846
                     0283   847 ;
                     0283   848 ; Save address of nonpaged listhead and run at IPL
                     0283   849 ; contained there.
                     0283   850 ;
                     0283   851
              52 00000000'EF DE 0283   852         MOVAL   EXE$GL_NONPAGED,R2      ; get nonpaged pool listhead
                     028A   853         DSBINT  (R2)+                   ; set ipl for pool access
                     0290   854
                     0290   855 ;
                     0290   856 ; Run through the LRP list and count the packets remaining
                     0290   857 ;
                     0290   858
              50 00000000'EF DE 0290   859         MOVAL   IOC$GL_LRPFL,R0         ; get LRP listhead address
                   51 50 D0 0297   860         MOVL    R0,R1                   ; copy header address
                     029A   861
                   51 61 D0 029A   862 10$:     MOVL    (R1),R1                 ; get forward link
                   50 51 D1 029D   863         CMPL    R1,R0                   ; point back to header?
                      04 13 02A0   864         BEQL    20$                     ; done if so
                      53 D6 02A2   865         INCL    R3                      ; count one more packet
                      F4 11 02A4   866         BRB     10$                     ; loop back for more
                     02A6   867
                     02A6   868 20$:     ENBINT                          ; enable interrupts
            0030'CF 53 D0 02A9   869 30$:     MOVL    R3,W^LRPCNT             ; save LRP count for FETCH rtn
                      04 02AE   870         RET
```

```
                                02AF    872          .SBTTL  PAGE_PRE - PAGE Class Pre-collection Rtn
                                02AF    873 ;++
                                02AF    874 ;
                                02AF    875 ; FUNCTIONAL DESCRIPTION:
                                02AF    876 ;
                                02AF    877 ;       This routine simply grabs the system page fault
                                02AF    878 ;       count and places it into a location accessible to
                                02AF    879 ;       the FETCH rtn.
                                02AF    880 ;
                                02AF    881 ; CALLING SEQUENCE:
                                02AF    882 ;
                                02AF    883 ;       CALLS/CALLG
                                02AF    884 ;
                                02AF    885 ; INPUTS:
                                02AF    886 ;
                                02AF    887 ;       4(AP) - address of current collection buffer (unused by this rtn)
                                02AF    888 ;
                                02AF    889 ; IMPLICIT INPUTS:
                                02AF    890 ;
                                02AF    891 ;       MMG$GL_SYSPHD - system process header address
                                02AF    892 ;
                                02AF    893 ; OUTPUTS:
                                02AF    894 ;
                                02AF    895 ;       None
                                02AF    896 ;
                                02AF    897 ; IMPLICIT OUTPUTS:
                                02AF    898 ;
                                02AF    899 ;       SYSFAULTS - contains accumulated total of system page faults
                                02AF    900 ;
                                02AF    901 ; ROUTINE VALUE:
                                02AF    902 ;
                                02AF    903 ;       R0 = SS$_NORMAL
                                02AF    904 ;
                                02AF    905 ;       R1 = YES, if subsequent FETCH collection is required.
                                02AF    906 ;       R1 = NO,  if subsequent FETCH collection is NOT required.
                                02AF    907 ;
                                02AF    908 ; SIDE EFFECTS:
                                02AF    909 ;
                                02AF    910 ;       none
                                02AF    911 ;--
                                02AF    912
                        0000    02AF    913 .ENTRY  PAGE_PRE, ^M<>
                                02B1    914
 50   00000000'EF      D0      02B1    915         MOVL    MMG$GL_SYSPHD,R0            ; get system header address
  0048'CF    4C A0    D0      02B8    916         MOVL    PHD$L_PAGEFLTS(R0),W^SYSFAULTS ; store system page fault count
                                02BE    917                                         ; for page display
                                02BE    918 ;
                                02BE    919 ; Indicate to caller that FETCH collection IS required.
                                02BE    920 ;
                                02BE    921
 51   00000000'8F      D0      02BE    922         MOVL    #YES,R1                     ; FETCH collection required
 50   00000000'8F      D0      02C5    923         MOVL    #SS$_NORMAL,R0              ; success status
                        04      02CC    924         RET                                 ; return
```

```
                                02CD   926          .SBTTL  STATES_PRE - STATES Class Pre-collection Rtn
                                02CD   927 ;++
                                02CD   928 ;
                                02CD   929 ; FUNCTIONAL DESCRIPTION:
                                02CD   930 ;
                                02CD   931 ;       Loop through all PCBs and count the number of processes in
                                02CD   932 ;       each scheduling state.  The counts are accumulated in the
                                02CD   933 ;       collection buffer passed to this rtn by the FETCH rtn.
                                02CD   934 ;
                                02CD   935 ; CALLING SEQUENCE:
                                02CD   936 ;
                                02CD   937 ;       CALLS/CALLG
                                02CD   938 ;
                                02CD   939 ; INPUTS:
                                02CD   940 ;
                                02CD   941 ;       4(AP) - address of current collection buffer (data portion)
                                02CD   942 ;
                                02CD   943 ; IMPLICIT INPUTS:
                                02CD   944 ;
                                02CD   945 ;       CDBPTR - global variable, pointer to current CDB
                                02CD   946 ;       SCH$GL_PCBVEC - contains address of PCB vector
                                02CD   947 ;       SCH$GL_MAXPIX - maximum process index
                                02CD   948 ;
                                02CD   949 ; OUTPUTS:
                                02CD   950 ;
                                02CD   951 ;       Collection buffer filled with appropriate state count values.
                                02CD   952 ;       OTHER_STATES and PROC_COUNT filled in for SYSTEM class.
                                02CD   953 ;
                                02CD   954 ; IMPLICIT OUTPUTS:
                                02CD   955 ;
                                02CD   956 ;       BARSIZE - global variable altered to indicate size of VT55
                                02CD   957 ;                 bar for histogram display.
                                02CD   958 ;
                                02CD   959 ; ROUTINE VALUE:
                                02CD   960 ;
                                02CD   961 ;       R0 = SS$_NORMAL
                                02CD   962 ;
                                02CD   963 ;       R1 = YES, if subsequent FETCH collection is required.
                                02CD   964 ;       R1 = NO,  if subsequent FETCH collection is NOT required.
                                02CD   965 ;
                                02CD   966 ; SIDE EFFECTS:
                                02CD   967 ;
                                02CD   968 ;       none
                                02CD   969 ;--
                                02CD   970
                        07FC    02CD   971 .ENTRY  STATES_PRE,     ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10>
                                02CF   972
                                02CF   973 ;
                                02CF   974 ; Reset counters in collection buffer to zero
                                02CF   975 ;
                                02CF   976
                          5A    D4     02CF   977          CLRL    R10                         ; clear counter for check of SYSTEM
                                       02D1   978                                              ; class state list
                       57   08    D0     02D1   979          MOVL    #SYSMGR_STATETOT,R7         ; store limit for state list to R7
                  56   00000000'EF    D0     02D4   980          MOVL    CDBPTR,R6                   ; Get STATES CDB ptr
04 BC    20 A6   00    6E    00    2C     02DB   981          MOVC5   #0,(SP),#0,CDB$W_BLKLEN(R6),@4(AP) ; zero collection buffer
                00000090'EF    D4     02E3   982          CLRL    PROC_COUNT                  ; Clear process count
```

```
                00000094'EF   D4  02E9   983          CLRL    OTHER_STATES            ; Clear cnt of processes in misc states
          55    00000000'EF   D0  02EF   984          MOVL    SCH$GL_MAXPIX,R5        ; get max number of processes
                              02F6   985
              0000'CF    0F   9A  02F6   986          MOVZBL  #15,W^BARSIZE           ; shrink bar size for VT55
          50    00000000'EF   D0  02FB   987          MOVL    SCH$GL_PCBVEC,R0        ; get address of PCB vector
                        51    D4  0302   988          CLRL    R1                      ; clear counter
                52    6041    D0  0304   989          MOVL    (R0)[R1],R2             ; get address of null process PCB
          53    04 AC   04    C3  0308   990          SUBL3   #4,4(AP),R3             ; address to put data ( states start at one)
                54      52    D0  030D   991          MOVL    R2,R4                   ; copy null PCB for first time
                        09    11  0310   992          BRB     20$                     ; skip null check first time through
                              0312   993 10$:
                54    6041    D0  0312   994          MOVL    (R0)[R1],R4             ; get next PCB address
                  52    54    D1  0316   995          CMPL    R4,R2                   ; does this point to null PCB?
                        27    13  0319   996          BEQL    30$                     ; try next one if so
                54  2C A4    3C  031B   997 20$:      MOVZWL  PCB$W_STATE(R4),R4      ; else get state number
                    6344    D6  031F   998          INCL    (R3)[R4]                ; incr counter for that state
                00000090'EF   D6  0322   999          INCL    PROC_COUNT              ; increment total process count
                              0328  1000
                              0328  1001 ;
                              0328  1002 ; Check to see if the state this process is in is one of those specified
                              0328  1003 ; in the SYSTEM class, and, if so, increment a counter (R10)
                              0328  1004 ;
                              0328  1005
                56    01      D0  0328  1006          MOVL    #1,R6                   ; init loop counter
          58    00000098'EF   DE  032B  1007          MOVAL   SYSMGR_STATES,R8        ; start of SYSTEM class state list
                59    88      9A  0332  1008 25$:     MOVZBL  (R8)+,R9                ; move state number to R9
                  54    59    D1  0335  1009          CMPL    R9,R4                   ; Compare it to the current state
                        04    12  0338  1010          BNEQ    27$                     ; branch if no match
                        5A    D6  033A  1011          INCL    R10                     ; found a match , increment count
                        04    11  033C  1012          BRB     30$                     ; Done with state check loop
              F0 56     57    F3  033E  1013 27$:     AOBLEQ  R7,R6,25$               ; continue until end of the list
                              0342  1014
                              0342  1015
              CC 51     55    F3  0342  1016 30$:     AOBLEQ  R5,R1,10$               ; continue until max index
                              0346  1017
                              0346  1018
                              0346  1019 ;
                              0346  1020 ; The total number of processes, minus the sum of processes in one of the
                              0346  1021 ; states explicitly specified in the SYSTEM class, equals the number of
                              0346  1022 ; processes in the OTHER category.
                              0346  1023 ;
                              0346  1024
00000094'EF     00000090'EF   5A  C3  0346  1025          SUBL3   R10,PROC_COUNT,OTHER_STATES
                              0352  1026
                              0352  1027 ;
                              0352  1028 ; Indicate to caller that FETCH collection is NOT required.
                              0352  1029 ;
                              0352  1030
                51    00000000'8F   D0  0352  1031          MOVL    #NO,R1                  ; FETCH collection NOT required
                50    00000000'8F   D0  0359  1032          MOVL    #SS$_NORMAL,R0          ; success status
                                    04  0360  1033          RET                             ; return
```

```
0361  1035          .SBTTL  MODES_PRE - MODES Class Pre-collection Rtn
0361  1036 ;++
0361  1037 ;
0361  1038 ; FUNCTIONAL DESCRIPTION:
0361  1039 ;
0361  1040 ;       Fetch and store the 6 mode counters for each processor
0361  1041 ;       (Interrupt, Kernel, Executive, Supervisor, User, Compat
0361  1042 ;       mode tick counters). Also, compute and store null time
0361  1043 ;       on each processor. Then adjust Primary Kernel and Secondary
0361  1044 ;       Interrupt times to remove the idle ticks contained in
0361  1045 ;       those counters.
0361  1046 ;
0361  1047 ; CALLING SEQUENCE:
0361  1048 ;
0361  1049 ;       CALLS/CALLG
0361  1050 ;
0361  1051 ; INPUTS:
0361  1052 ;
0361  1053 ;       4(AP) - address of current collection buffer (data portion)
0361  1054 ;
0361  1055 ; IMPLICIT INPUTS:
0361  1056 ;
0361  1057 ;       SCH$GL_PCBVEC  - contains address of PCB vector
0361  1058 ;
0361  1059 ; OUTPUTS:
0361  1060 ;
0361  1061 ;       None
0361  1062 ;
0361  1063 ; IMPLICIT OUTPUTS:
0361  1064 ;
0361  1065 ;       Collection buffer filled with 7 (or 14, if multiprocessor)
0361  1066 ;       mode counter values. The values are fetched directly from
0361  1067 ;       the system, with the exception of:
0361  1068 ;
0361  1069 ;               Primary Kernel
0361  1070 ;               Primary Null
0361  1071 ;               Secondary Interrupt
0361  1072 ;               Secondary Null
0361  1073 ;
0361  1074 ;       These values are calculated as follows. Pick up Secondary
0361  1075 ;       Null from MPS$GL_NULLCPU. Re-compute Secondary Interrupt
0361  1076 ;       by subtracting Secondary Null from it. Compute Primary Null
0361  1077 ;       by subtracting Secondary Null from NULL PHD CPUTIM. Finally,
0361  1078 ;       re-compute Primary Kernel by subtracting Primary Null from it.
0361  1079 ;
0361  1080 ; ROUTINE VALUE:
0361  1081 ;
0361  1082 ;       R0 = SS$_NORMAL
0361  1083 ;
0361  1084 ;       R1 = YES, if subsequent FETCH collection is required.
0361  1085 ;       R1 = NO,  if subsequent FETCH collection is NOT required.
0361  1086 ;
0361  1087 ; SIDE EFFECTS:
0361  1088 ;
0361  1089 ;       None
0361  1090 ;--
```

```
                     001C  0361  1092 .ENTRY  MODES_PRE, ^M<R2,R3,R4>
                           0363  1093
                     54    D4    0363  1094          CLRL    R4                      ; assume no Secondary null time
                52   04 AC   D0    0365  1095          MOVL    4(AP),R2                ; get pointer to coll buff (data portion)
           53   00000000'GF   DE    0369  1096          MOVAL   G^PMS$GL_KERNEL,R3      ; get ptr to Primary mode counters
                           0370  1097
                           0370  1098 ;
                           0370  1099 ; Load collection buffer with Primary mode counters
                           0370  1100 ;
                           0370  1101
                           0370  1102 10$:
                82   10 A3   D0    0370  1103          MOVL    <4*4>(R3),(R2)+         ; Interrupt
                     82   63   7D    0374  1104          MOVQ    (R3),(R2)+              ; Kernel, Exec
                82   08 A3   7D    0377  1105          MOVQ    <2*4>(R3),(R2)+         ; Supervisor, User
                82   14 A3   D0    037B  1106          MOVL    <5*4>(R3),(R2)+         ; Compat
           51   00000000'FF   D0    037F  1107          MOVL    @SCH$GL_PCBVEC,R1       ; get null pcb address
                51   6C A1   D0    0386  1108          MOVL    PCB$L_PHD(R1),R1        ; get null phd address
                62   38 A1   D0    038A  1109          MOVL    PHD$L_CPUTIM(R1),(R2)   ; get idle time on Primary
                           038E  1110
                           038E  1111 ;
                           038E  1112 ; Load collection buffer with Secondary mode counters
                           038E  1113 ;
                           038E  1114
           51   00000000'EF   D0    038E  1115          MOVL    SPTR,R1                 ; load SYI pointer
                01   0D A1   91    0395  1116          CMPB    MNR_SYI$B_MPCPUS(R1),#1 ; just one processor?
                     60   13    0399  1117          BEQL    50$                     ; yes -- skip Secondary processing
                           039B  1118
            7E   04 AC    1C    C1    039B  1119          ADDL3   #<7*4>,4(AP),-(SP)      ; push addr of Secondary coll buff
                     01   DD    03A0  1120          PUSHL   #1                      ; push argument count
                51   5E   D0    03A2  1121          MOVL    SP,R1                   ; save arg list address
                           03A5  1122          $CMKRNL_S  W^GETSEC,(R1)        ; get secondary ctrs into coll buff
                           03B2  1123
                52   04 AC   D0    03B2  1124          MOVL    4(AP),R2                ; re-instate collection buffer ptr
                54   34 A2   D0    03B6  1125          MOVL    <13*4>(R2),R4           ; save Secondary null for use below
                           03BA  1126
                           03BA  1127 ;
                           03BA  1128 ; Establish new BASE counters if necessary
                           03BA  1129 ;
                           03BA  1130
                2B 50   E9    03BA  1131          BLBC    R0,30$                  ; br if no need to estab new base
                           03BD  1132
                           03BD  1133 ;
                           03BD  1134 ; Get pointer to Secondary counters from PREVIOUS collection buffer
                           03BD  1135 ;
                           03BD  1136
           51   00000000'EF   D0    03BD  1137          MOVL    CDBPTR,R1               ; get MODES CDB pointer
                52   2E A1   D0    03C4  1138          MOVL    CDB$A_BUFFERS(R1),R2    ; get buffer block pointer
                     53   62   D0    03C8  1139          MOVL    MBP$A_BUFFERA(R2),R3    ; assume buffer A is PREVIOUS
           04 4B A1   01    E0    03CB  1140          BBS     #CDB$V_SWAPBUF,CDB$L_FLAGS(R1),20$ ; branch if so
                53   04 A2   D0    03D0  1141          MOVL    MBP$A_BUFFERB(R2),R3    ; else load buffer B ptr
                           03D4  1142 20$:
                     53   29   C0    03D4  1143          ADDL2   #<MNR_CLS$K_HSIZE+<7*4>>,R3 ; point to counters
                           03D7  1144
                52   0074'CF   DE    03D7  1145          MOVAL   W^BASE,R2               ; get ptr to base counters
                     82   83   7D    03DC  1146          MOVQ    (R3)+,(R2)+             ; establish new base
                     82   83   7D    03DF  1147          MOVQ    (R3)+,(R2)+             ; ....
                     82   83   7D    03E2  1148          MOVQ    (R3)+,(R2)+             ; ....
```

```
                62   63   D0   03E5  1149          MOVL    (R3),(R2)               ; ....
```

```
                        03E8  1151 ;
                        03E8  1152 ; Add BASE counter values to collection buffer
                        03E8  1153 ;
                        03E8  1154
                        03E8  1155 30$:
    53    0074'CF   DE  03E8  1156          MOVAL   W^BASE,R3                ; address of BASE counters
 52   04 AC    1C   C1  03ED  1157          ADDL3   #<7*4>,4(AP),R2          ; compute addr of coll buff ctrs
          51    07   D0  03F2  1158          MOVL    #7,R1                    ; load number of counters
                        03F5  1159 40$:
          B2    83   C0  03F5  1160          ADDL2   (R3)+,(R2)+              ; add BASE ctr value to coll buff
            FA  51   F5  03F8  1161          SOBGTR  R1,40$                   ; loop for each counter
                        03FB  1162
                        03FB  1163 ;
                        03FB  1164 ; Compute Primary Kernel time and Primary Null time
                        03FB  1165 ;
                        03FB  1166
                        03FB  1167 50$:
       52    04 AC   D0  03FB  1168          MOVL    4(AP),R2                 ; re-instate collection buffer ptr
       18 A2    54   C2  03FF  1169          SUBL2   R4,<6*4>(R2)             ; compute null time on Primary
    04 A2    18 A2   C2  0403  1170          SUBL2   <6*4>(R2),<1*4>(R2)      ; subtract it from Primary kernel mode
                        0408  1171
51    00000000'8F   D0  0408  1172          MOVL    #NO,R1                   ; indicate FETCH collection NOT required
50    00000000'8F   D0  040F  1173          MOVL    #SS$_NORMAL,R0           ; success status
                    04  0416  1174          RET                              ; return
```

```
0417  1176 ;++
0417  1177 ; GETSEC - Routine to get Secondary processor mode counters
0417  1178 ;
0417  1179 ; CALLING SEQUENCE:
0417  1180 ;
0417  1181 ;       $CMKRNL_S  GETSEC,arglist_addr
0417  1182 ;
0417  1183 ; INPUTS:
0417  1184 ;
0417  1185 ;       4(AP) - address of Secondary portion of CURRENT collection buffer
0417  1186 ;
0417  1187 ; OUTPUTS:
0417  1188 ;
0417  1189 ;       None
0417  1190 ;
0417  1191 ; IMPLICIT INPUTS:
0417  1192 ;
0417  1193 ;       EXE$GL_MP       - contains address of multiprocessing code
0417  1194 ;       MPS$AL_CPUTIME - contains address of Secondary mode counters
0417  1195 ;       MPS$GL_NULLCPU - contains count of Secondary null ticks
0417  1196 ;       MPS$GQ_MPSTRTIM - quadword time at which MP code loaded
0417  1197 ;       MPSTRTIM -        MPS$GQ_MPSTRTIM value at previous interval
0417  1198 ;       MCA$A_MPADDR -    EXE$GL_MP value at previous interval
0417  1199 ;
0417  1200 ; IMPLICIT OUTPUTS:
0417  1201 ;
0417  1202 ;       Secondary portion of CURRENT collection buffer is filled
0417  1203 ;
0417  1204 ; ROUTINE VALUE:
0417  1205 ;
0417  1206 ;       R0 = YES, if loading of new BASE counters is required.
0417  1207 ;       R0 = NO,  if loading of new BASE counters is NOT required.
0417  1208 ;
0417  1209 ; SIDE EFFECTS:
0417  1210 ;
0417  1211 ;       Must raise IPL to synchronize database access
0417  1212 ;--
```

```
                                0417  1214 GETSEC:
                        0FFC    0417  1215          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                0419  1216
                     55   7C    0419  1217          CLRQ    R5                      ; clear Secondary mode counter regs
                     57   7C    041B  1218          CLRQ    R7                      ; ....
                     59   7C    041D  1219          CLRQ    R9                      ; ....
                     5B   D4    041F  1220          CLRL    R11                     ; ....
                                0421  1221
                                0421  1222 ;
                                0421  1223 ; Pick up all data needed from MP data structures at IPL SYNCH
                                0421  1224 ;
                                0421  1225
                                0421  1226 10$:     SETIPL  30$                     ; Raise IPL (and lock pages in w.s.)
                50   00000000'GF   D0  0428  1227          MOVL    G^EXE$GL_MP,R0          ; get ptr to MP code
                           1E   13  042F  1228          BEQL    20$                     ; br if not there
                5B    0000'C0   D0  0431  1229          MOVL    MPS$GL_NULLCPU(R0),R11  ; get Secondary null time
                54    0000'C0   9E  0436  1230          MOVAB   MPS$AL_CPUTIME(R0),R4   ; get ptr to Secondary mode counters
                                0438  1231
                                0438  1232 ;
                                0438  1233 ; Get Secondary mode counters
                                0438  1234 ;
                                0438  1235
                  55    10 A4   D0  0438  1236          MOVL    <4*4>(R4),R5            ; Interrupt
                     56    64   7D  043F  1237          MOVQ    (R4),R6                 ; Kernel, Exec
                  58    08 A4   7D  0442  1238          MOVQ    <2*4>(R4),R8            ; Supervisor, User
                  5A    14 A4   D0  0446  1239          MOVL    <5*4>(R4),R10           ; Compat
                                044A  1240
                52    0000'C0   7D  044A  1241          MOVQ    MPS$GQ_MPSTRTIM(R0),R2  ; get MP start time
                                044F  1242
                                044F  1243 20$:     SETIPL  #0                      ; lower IPL
                        04      11  0452  1244          BRB     40$                     ; branch around data
                                0454  1245
                00000008        0454  1246 30$:     .LONG   IPL$_SYNCH
                                0458  1247          ASSUME  .-10$  LE  512          ; Make sure it doesn't exceed two pages
```

```
                                0458  1249 ;
                                0458  1250 ; Move counter registers into CURRENT collection buffer
                                0458  1251 ;
                                0458  1252
                                0458  1253 40$:
          55    5B    C2        0458  1254          SUBL2   R11,R5                  ; compute Secondary interrupt time
                                045B  1255                                          ; (by subtracting out null time)
     54   04 AC    D0           045B  1256          MOVL    4(AP),R4                ; get addr of Secondary coll buff
        84    55    7D          045F  1257          MOVQ    R5,(R4)+                ; move in the counter values
        84    57    7D          0462  1258          MOVQ    R7,(R4)+                ; ....
        84    59    7D          0465  1259          MOVQ    R9,(R4)+                ; ....
        64    5B    D0          0468  1260          MOVL    R11,(R4)                ; ....
                                046B  1261
                                046B  1262 ;
                                046B  1263 ; Determine if new BASE counters have to be established
                                046B  1264 ;
                                046B  1265
51   00000000'EF    D0          046B  1266          MOVL    MCAPTR,R1               ; get MCA pointer
          1C A1    D5          0472  1267          TSTL    MCA$A_MPADDR(R1)        ; was MP running at last interval?
                1D 13          0475  1268          BEQL    60$                     ; no -- don't need new BASE
      1C A1    50  D0          0477  1269          MOVL    R0,MCA$A_MPADDR(R1)     ; save MP addr for this interval
                0E 13          047B  1270          BEQL    50$                     ; if 0 now, need new BASE
     53   0070'CF  D1          047D  1271          CMPL    W^MPSTRTIM+4,R3         ; has MP start time changed?
                07 12          0482  1272          BNEQU   50$                     ; yes -- need new BASE
     52   006C'CF  D1          0484  1273          CMPL    W^MPSTRTIM,R2           ; check the other half of time
                09 13          0489  1274          BEQLU   60$                     ; no change -- don't need new base
                               048B  1275 50$:
50   00000000'8F    D0          048B  1276          MOVL    #YES,R0                 ; indicate new BASE ctr values needed
                08 11          0492  1277          BRB     70$                     ; go return
                               0494  1278 60$:
      1C A1    50  D0          0494  1279          MOVL    R0,MCA$A_MPADDR(R1)     ; save MP addr for this interval
50   00000000'8F    D0          0498  1280          MOVL    #NO,R0                  ; indicate new BASE values not needed
                               049F  1281 70$:
     006C'CF  52    7D          049F  1282          MOVQ    R2,W^MPSTRTIM           ; save new MP start time
                04 04A4  1283          RET
```

```
                                                C 4
                                .SBTTL  PROC_PRE - PROCESSES Class Pre-collection Rtn
                                ;++
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       Loop through all PCBs and collect information on each
                                ;       process, as well as the process count. The info is stored
                                ;       in the collection buffer passed to this rtn by the FETCH rtn.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       CALLS/CALLG
                                ;
                                ; INPUTS:
                                ;
                                ;       4(AP) - address of current collection buffer (data portion)
                                ;
                                ; IMPLICIT INPUTS:
                                ;
                                ;       None
                                ;
                                ; OUTPUTS:
                                ;
                                ;       None
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;
                                ;       Collection buffer filled with data for each process.
                                ;
                                ; ROUTINE VALUE:
                                ;
                                ;       R0 = SS$_NORMAL
                                ;
                                ;       R1 = YES, if subsequent FETCH collection is required.
                                ;       R1 = NO,  if subsequent FETCH collection is NOT required.
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       none
                                ;--
                                ;
0000                            .ENTRY  PROC_PRE, ^M<>
                                ;
                                        $CMKRNL_S B^SCANPROCS,(AP)      ; Scan all processes in kernel mode
                                ;
                                ; Indicate to caller that FETCH collection is NOT required.
                                ;
                                ;
51  00000000'8F   D0            MOVL    #NO,R1                          ; FETCH collection NOT required
50  00000000'8F   D0            MOVL    #SS$_NORMAL,R0                  ; success status
              04    RET                                                 ; Return
```

```
                                                     ;++
                                                     ; SCANPROCS - subroutine to scan processes in kernel mode
                                                     ;
                                                     ; CALLING SEQUENCE:
                                                     ;
                                                     ;       $CMKRNL_S SCANPROCS,(AP)
                                                     ;
                                                     ; IMPLICIT INPUTS:
                                                     ;
                                                     ;       SCH$GL_PCBVEC - contains address of PCB vector
                                                     ;       SCH$GL_MAXPIX - maximum process index
                                                     ;
                                                     ; IMPLICIT OUTPUTS:
                                                     ;
                                                     ;       Collection buffer filled with data for each process.
                                                     ;
                                                     ; SIDE EFFECTS:
                                                     ;
                                                     ;       Some of this routine is executed at IPL SYNCH to synchronize
                                                     ;       the use of the PCB Vector and the PHD for each process.
                                                     ;--
                                                     ;
                                                     SCANPROCS:
                                 OFFC                .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Register save mask
                                                     ;
      54    04 AC    08    C1                       ADDL3   #MNR_PRO$K_PSIZE,4(AP),R4 ; Point past the prefix to ...
                                                     ;                                 ; ... beginning of data blocks
                       55    D4                       CLRL    R5                        ; Clear process counter
      52    00000000'EF    D0                       MOVL    SCH$GL_PCBVEC,R2          ; Point to top of PCB vector
                  50    62    D0                       MOVL    (R2),R0                   ; Get NULL PCB address
                57    60 A0    D0                       MOVL    PCB$L_PID(R0),R7          ; ... and its PID
                   56    50    D0                       MOVL    R0,R6                     ; Remember NULL PCB address
                         53    D4                       CLRL    R3                        ; Clear current pix
                         1A    11                       BRB     30$                       ; Jump into loop to collect the NULL process
                                                     ;
                                                     10$:
                                                     ;
                                                     SETIPL  80$                       ; Synchronize use of PCB vector
                  50    6243    D0                       MOVL    (R2)[R3],R0               ; Get next PCB address
                57    60 A0    D0                       MOVL    PCB$L_PID(R0),R7          ; ... and its PID
                                                     SETIPL  #0                        ; Back to IPL 0
                                                     ;
                   50    56    D1                       CMPL    R6,R0                     ; Is this an empty slot (= NULL PCB)?
                         03    12                       BNEQ    30$                       ; No -- go collect it
                      008C    31                       BRW     70$                       ; Yes -- skip collection
                                                     ;
                                                     30$:
                                                     ;
      64    60 A0    D0                       MOVL    PCB$L_PID(R0),    MNR_PRO$L_IPID(R4)      ; Move PCB items
04 A4    00BC C0    D0                       MOVL    PCB$L_UIC(R0),    MNR_PRO$L_UIC(R4)       ; ... into
  08 A4    2C A0    B0                       MOVW    PCB$W_STATE(R0),  MNR_PRO$W_STATE(R4)     ; ... collection
  0A A4    0B A0    90                       MOVB    PCB$B_PRI(R0),    MNR_PRO$B_PRI(R4)       ; ... buffer
  0B A4    70 A0    7D                       MOVQ    PCB$T_LNAME(R0),  MNR_PRO$Q_LNAME(R4)     ; 1st half of p name
  13 A4    78 A0    7D                       MOVQ    PCB$T_LNAME+8(R0),MNR_PRO$Q_LNAME+8(R4)   ; ... second half
  1B A4    34 A0    B0                       MOVW    PCB$W_GPGCNT(R0), MNR_PRO$W_GPGCNT(R4)    ;
  1D A4    36 A0    B0                       MOVW    PCB$W_PPGCNT(R0), MNR_PRO$W_PPGCNT(R4)    ;
  33 A4    64 A0    D0                       MOVL    PCB$L_EPID(R0),   MNR_PRO$L_EPID(R4)      ;
  37 A4    4C A0    D0                       MOVL    PCB$L_EFWM(R0),   MNR_PRO$L_EFWM(R4)      ;
                                      052C
```

```
                     052C  1395          SETIPL  #0$                     ; Synchronize use of PCB vector
   51    57    3C    0533  1396          MOVZWL  R7,R1                   ; Turn PID into PCB vector index
   51    6241  D0    0536  1397          MOVL    (R2)[R1],R1             ; Get PCB address
57    60  A1  D1    053A  1398          CMPL    PCB$L_PID(R1),R7        ; Check to see if PID is still the same
         05    13    053E  1399          BEQLU   40$                     ; Continue if so
                     0540  1400          SETIPL  #0                      ; Otherwise, return to IPL 0,
         41    11    0543  1401          BRB     70$                     ; ... and skip this process
                     0545  1402
                     0545  1403 40$:
57    24  A0  D0    0545  1404          MOVL    PCB$L_STS(R0),R7        ; Save status field while SYNCHed
09 57     00  E0    0549  1405          BBS     #PCB$V_RES,R7,50$       ; If process resident, go after PHD info
                     054D  1406          SETIPL  #0                      ; Otherwise, return to IPL 0,
         58    7C    0550  1407          CLRQ    R8                      ; ... indicate no PHD statistics
         5A    7C    0552  1408          CLRQ    R10                     ; ...
         17    11    0554  1409          BRB     60$                     ; ... and continue
                     0556  1410
                     0556  1411 50$:
51    6C  A0  D0    0556  1412          MOVL    PCB$L_PHD(R0),R1        ; Get PHD address
58    54  A1  D0    055A  1413          MOVL    PHD$L_DIOCNT(R1),R8     ; Get PHD stats while still at raised IPL
59    4C  A1  D0    055E  1414          MOVL    PHD$L_PAGEFLTS(R1),R9   ; Use registers to avoid page faults
5A    38  A1  D0    0562  1415          MOVL    PHD$L_CPUTIM(R1),R10    ; ...
5B    58  A1  D0    0566  1416          MOVL    PHD$L_BIOCNT(R1),R11    ; ...
                     056A  1417          SETIPL  #0                      ; Back to IPL 0
                     056D  1418
                     056D  1419 60$:
1F A4     57    D0    056D  1420          MOVL    R7,MNR_PRO$L_STS(R4)    ; Status field into collection buffer
23 A4     58    D0    0571  1421          MOVL    R8,MNR_PRO$L_DIOCNT(R4) ; Four PHD fields into collection buffer
27 A4     59    D0    0575  1422          MOVL    R9,MNR_PRO$L_PAGEFLTS(R4) ; ...
2B A4     5A    D0    0579  1423          MOVL    R10,MNR_PRO$L_CPUTIM(R4) ; ...
2F A4     5B    D0    057D  1424          MOVL    R11,MNR_PRO$L_BIOCNT(R4) ; ...
                     0581  1425
         55    D6    0581  1426          INCL    R5                      ; Count this process
      54   3B    C0    0583  1427          ADDL2   #MNR_PRO$K_DSIZE,R4     ; ... and point to next data block in buffer
                     0586  1428                                          ; NOTE -- OK to use the MNR_PRO$K_DSIZE
                     0586  1429                                          ; ... constant, since live collection
                     0586  1430
                     0586  1431 70$:
FF50 53   01  00000000'EF  F1    0586  1432          ACBL    5CH$GL_MAXPIX,#1,R3,10$ ; Loop once for each process in PCBVEC
      51    04  AC  D0    0590  1433          MOVL    4(AP),R1                ; Point to prefix portion of coll buffer
         61    55  D0    0594  1434          MOVL    R5,MNR_PRO$L_PCTREC(R1) ; Move # of procs this record into buffer
      04  A1    55  D0    0597  1435          MOVL    R5,MNR_PRO$L_PCTINT(R1) ; Move # of procs this interval into buffer
                  04    059B  1436          RET                             ; Return to EXEC mode for exit
                     059C  1437
                00000008  059C  1438 80$:     .LONG   IPL$_SYNCH
                     05A0  1439          ASSUME  .-10$  LE  512          ; Make sure it doesn't exceed two pages
```

```
                        05A0  1441          .SBTTL  DISK_PRE - DISK Class Pre-collection Rtn
                        05A0  1442 ;++
                        05A0  1443 ;
                        05A0  1444 ; FUNCTIONAL DESCRIPTION:
                        05A0  1445 ;
                        05A0  1446 ;       Loop through entire device data base, collecting info on
                        05A0  1447 ;       each disk device.The info is stored in the collection buffer
                        05A0  1448 ;       passed to this rtn by the FETCH rtn.
                        05A0  1449 ;
                        05A0  1450 ; CALLING SEQUENCE:
                        05A0  1451 ;
                        05A0  1452 ;       CALLS/CALLG
                        05A0  1453 ;
                        05A0  1454 ; INPUTS:
                        05A0  1455 ;
                        05A0  1456 ;       4(AP) - address of current collection buffer (data portion)
                        05A0  1457 ;
                        05A0  1458 ; IMPLICIT INPUTS:
                        05A0  1459 ;
                        05A0  1460 ;       None
                        05A0  1461 ;
                        05A0  1462 ; OUTPUTS:
                        05A0  1463 ;
                        05A0  1464 ;       None
                        05A0  1465 ;
                        05A0  1466 ; IMPLICIT OUTPUTS:
                        05A0  1467 ;
                        05A0  1468 ;       Collection buffer filled with data for each disk.
                        05A0  1469 ;
                        05A0  1470 ; ROUTINE VALUE:
                        05A0  1471 ;
                        05A0  1472 ;       R0 = status from SCANDISKS routine
                        05A0  1473 ;
                        05A0  1474 ;       R1 = YES, if subsequent FETCH collection is required.
                        05A0  1475 ;       R1 = NO,  if subsequent FETCH collection is NOT required.
                        05A0  1476 ;
                        05A0  1477 ; SIDE EFFECTS:
                        05A0  1478 ;
                        05A0  1479 ;       none
                        05A0  1480 ;--
                        05A0  1481
                  0000  05A0  1482 .ENTRY  DISK_PRE, ^M<>
                        05A2  1483
                        05A2  1484          $CMKRNL_S B^SCANDISKS,(AP)      ; Scan all disk structs in kernel mode
                        05AE  1485
                        05AE  1486 ;
                        05AE  1487 ; Indicate to caller that FETCH collection is NOT required.
                        05AE  1488 ;
                        05AE  1489
 51   00000000'8F   D0  05AE  1490          MOVL    #NO,R1                  ; FETCH collection NOT required
                     04  05B5  1491          RET                             ; Return with status from SCANDISKS
```

```
                        05B6  1493 ;++
                        05B6  1494 ;
                        05B6  1495 ; SCANDISKS - subroutine to scan disk data structures in kernel mode
                        05B6  1496 ;
                        05B6  1497 ; CALLING SEQUENCE:
                        05B6  1498 ;
                        05B6  1499 ;       $CMKRNL_S SCANDISKS,(AP)
                        05B6  1500 ;
                        05B6  1501 ; INPUTS:
                        05B6  1502 ;
                        05B6  1503 ;       4(AP) - address of current collection buffer (data portion)
                        05B6  1504 ;
                        05B6  1505 ; OUTPUTS:
                        05B6  1506 ;
                        05B6  1507 ;       None
                        05B6  1508 ;
                        05B6  1509 ; IMPLICIT INPUTS:
                        05B6  1510 ;
                        05B6  1511 ;       SCH$LOCKR, SCH$UNLOCK - I/O Mutex lock and unlock routines.
                        05B6  1512 ;       IOC$SCAN_IODB         - Routine which scans the I/O data base
                        05B6  1513 ;                               for the next device/unit.
                        05B6  1514 ;       SCH$GL_CURPCB         - Current PCB.
                        05B6  1515 ;
                        05B6  1516 ; IMPLICIT OUTPUTS:
                        05B6  1517 ;
                        05B6  1518 ;       Collection buffer filled with data for each disk.
                        05B6  1519 ;
                        05B6  1520 ; ROUTINE VALUE:
                        05B6  1521 ;
                        05B6  1522 ;       R0 = SS$_NORMAL, or system service error status
                        05B6  1523 ;
                        05B6  1524 ; SIDE EFFECTS:
                        05B6  1525 ;
                        05B6  1526 ;       This routine holds the IO MUTEX and runs at ASTDEL IPL while
                        05B6  1527 ;       it is scanning the device data base.
                        05B6  1528 ;--
                        05B6  1529
                        05B6  1530 SCANDISKS:
                  0FD4  05B6  1531         .WORD   ^M<R2,R4,R6,R7,R8,R9,R10,R11> ; Register save mask
                        05B8  1532
 59   04 AC    08   C1  05B8  1533         ADDL3   #MNR_HOM$K_PSIZE,4(AP),R9 ; Point past the prefix to ...
                        05BD  1534                                           ; ... beginning of data blocks
                     58 D4  05BD  1535         CLRL    R8                    ; Clear disk counter
54   00000000'GF    DO   05BF  1536         MOVL    G^SCH$GL_CURPCB,R4    ; Get PCB for IOLOCKR call
     00000000'GF    16   05C6  1537         JSB     G^SCH$IOLOCKR         ; Get mutex to lock I/O data base
                        05CC  1538                                           ; NOTE -- now at IPL ASTDEL, so can
                        05CC  1539                                           ; ... take page faults
                        05CC  1540 ;
                        05CC  1541 ; Call IOC$SCAN_IODB to get the next unit in the I/O data base.
                        05CC  1542 ; The unit is described by the DDB and UCB pointers in R11 and
                        05CC  1543 ; R10, respectively. To begin the scan, call SCAN_IODB with R11
                        05CC  1544 ; and R10 containing zero. It returns the first unit in the data
                        05CC  1545 ; base in the same registers. On subsequent calls, simply leave
                        05CC  1546 ; R11 and R10 alone, and SCAN_IODB will return the next unit.
                        05CC  1547 ; If an entire DDB is undesireable, clear R10 before calling
                        05CC  1548 ; and all units for that device will be skipped.
                        05CC  1549 ;
```

```
                                05CC  1550
                        5B  D4  05CC  1551          CLRL    R11                     ; Indicate starting at beginning
                        5A  D4  05CE  1552          CLRL    R10                     ; ... of I/O data base
                                05D0  1553 10$:
            00000000'GF  16     05D0  1554          JSB     G^IOC$SCAN_IODB         ; Get the next unit
                  64  50  E9     05D6  1555          BLBC    R0,100$                 ; Br if at end of data base
                                05D9  1556
                                05D9  1557 ;
                                05D9  1558 ; Check the class of the device/unit just provided to see if we want it.
                                05D9  1559 ;
                                05D9  1560
                                05D9  1561 ;
                                05D9  1562 ; Check entire controller (DDB) for disk class by examining the UCB.
                                05D9  1563 ; If the DDB class is not disk, then clear R10 and branch back to get next
                                05D9  1564 ; device/unit. If it is disk, simply continue.
                                05D9  1565 ;
                                05D9  1566
             40 AA  01  91      05D9  1567          CMPB    #DC$_DISK,UCB$B_DEVCLASS(R10) ; Is the unit a disk?
                    04  13      05DD  1568          BEQL    20$                     ; Yes -- go check some more
                    5A  D4      05DF  1569          CLRL    R10                     ; No -- skip entire controller
                    ED  11      05E1  1570          BRB     10$                     ; Go get next one
                                05E3  1571
                                05E3  1572 ;
                                05E3  1573 ; Check for special class driver path UCB, and throw it out.
                                05E3  1574 ;
                                05E3  1575
                                05E3  1576 20$:
          E8 3C AA  03  E0      05E3  1577          BBS     #DEV$V_CDP,UCB$L_DEVCHAR2(R10),10$
                                05E8  1578                                          ; Skip UCB if class driver path
                                05E8  1579
                                05E8  1580 ;
                                05E8  1581 ; Check to see if disk is mounted, and throw out if not.
                                05E8  1582 ;
                                05E8  1583
        E3 38 AA  13  E1        05E8  1584          BBC     #DEV$V_MNT,UCB$L_DEVCHAR(R10),10$
                                05ED  1585                                          ; Skip UCB if not mounted
                                05ED  1586
                                05ED  1587 ;
                                05ED  1588 ; R11/R10 now point to a disk DDB/UCB. Collect pertinent data.
                                05ED  1589 ;
                                05ED  1590
             89  3C AB  90      05ED  1591          MOVB    DDB$L_ALLOCLS(R11),(R9)+ ; Collect allocation class
             89  14 AB  D0      05F1  1592          MOVL    DDB$T_NAME(R11),(R9)+   ; Collect the device name
             89  54 AA  B0      05F5  1593          MOVW    UCB$W_UNIT(R10),(R9)+   ; Collect the (binary) unit number
                                05F9  1594
             50  34 AB  D0      05F9  1595          MOVL    DDB$L_SB(R11),R0        ; Get system block pointer
                    04  12      05FD  1596          BNEQU   30$                     ; Br if there is one
                    89  7C      05FF  1597          CLRQ    (R9)+                   ; Else null node name
                    04  11      0601  1598          BRB     40$
                                0603  1599 30$:
             89  44 A0  7D      0603  1600          MOVQ    SB$T_NODENAME(R0),(R9)+ ; Collect the node name
                                0607  1601 40$:
             50  34 AA  D0      0607  1602          MOVL    UCB$L_VCB(R10),R0       ; Get VCB pointer
                    17  12      060B  1603          BNEQU   50$                     ; Br if there is one
          89    000000BC'EF  D0 060D  1604          MOVL    BLANKS,(R9)+            ; Else blank volume name
          89    000000BC'EF  D0 0614  1605          MOVL    BLANKS,(R9)+            ; ....
          89    000000BC'EF  D0 061B  1606          MOVL    BLANKS,(R9)+            ; ....
```

```
                08   11  0622  1607                BRB     60$
                                0624  1608  50$:
                89   14 A0  7D  0624  1609                MOVQ    VCBST_VOLNAME(R0),(R9)+ ; Collect the volume name
                89   1C A0  D0  0628  1610                MOVL    VCBST_VOLNAME+8(R0),(R9)+ ; ....
                                062C  1611  60$:
                89   70 AA  D0  062C  1612                MOVL    UCB$L_OPCNT(R10),(R9)+  ; Collect the operation count
                89   6A AA  32  0630  1613                CVTWL   UCB$W_QLEN(R10),(R9)+   ; Collect the queue length
                     03     18  0634  1614                BGEQ    70$                     ; Br if pos or zero (as expected)
                     FC A9  D4  0636  1615                CLRL    -4(R9)                  ; Clear it if negative
                                0639  1616                                                ; NOTE -- this is a transient condition,
                                0639  1617                                                ;   which clears itself on next coll'n
                                0639  1618  70$:
                                0639  1619
                                0639  1620  ;
                                0639  1621  ; ****JNL**** Start here.
                                0639  1622  ; *** NOTE *** The following lines of code which collect the journaling
                                0639  1623  ;              I/O operation count are temporarily commented out.
                                0639  1624  ;
                                0639  1625
                                0639  1626  ;
                                0639  1627  ; Collect the journaling I/O operation count for this unit
                                0639  1628  ;
                                0639  1629  ;
                                0639  1630  ;       CLRL    (R9)+                   ; Assume no journaling I/O
                                0639  1631  ;       MOVL    UCB$L_VCB(R10),R0       ; Get VCB pointer
                                0639  1632  ;       BEQL    90$                     ; Br if no VCB
                                0639  1633  ;       MOVL    VCB$L_JNLIOCNT(R0),-4(R9) ; Collect journaling I/O op count
                                0639  1634  ;90$:
                                0639  1635  ; ****JNL**** End here.
                                0639  1636  ;
                                0639  1637
                58   D6         0639  1638          INCL    R8                      ; Count this unit
                93   11         063B  1639          BRB     10$                     ; Go get next device/unit
                                063D  1640
                                063D  1641  ;
                                063D  1642  ; The entire I/O data base has been scanned. Relinquish the I/O Mutex
                                063D  1643  ; and drop IPL back to 0.
                                063D  1644  ;
                                063D  1645
                                063D  1646  100$:
          54    00000000'GF  D0  063D  1647          MOVL    G^SCH$GL_CURPCB,R4      ; Get PCB for IOUNLOCK call
                00000000'GF  16  0644  1648          JSB     G^SCH$IOUNLOCK          ; Relinquish lock on I/O data base
                                064A  1649                                          ; NOTE -- this rtn clobbers R0-R2
                                064A  1650          SETIPL  #0                      ; Return to IPL 0
                                064D  1651
                50   04 AC  D0  064D  1652          MOVL    4(AP),R0                ; Point to prefix part of coll buff
                   60   58  D0  0651  1653          MOVL    R8,MNR_HOM$L_ELTCT(R0)  ; Save element count
                     04 A0  D4  0654  1654          CLRL    MNR_HOM$L_RESERVED(R0)  ; Clear reserved longword
          50    00000000'8F  D0  0657  1655          MOVL    #SS$_NORMAL,R0          ; Success status
                               04  065E  1656          RET                             ; Return with status
```

```
                                        .SBTTL  JDEVICE_PRE - JDEVICE Class Pre-collection Rtn
                                ;++
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       Loop through entire device data base, collecting info on
                                ;       each journal device. The info is stored in the collection buffer
                                ;       passed to this rtn by the FETCH rtn.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       CALLS/CALLG
                                ;
                                ; INPUTS:
                                ;
                                ;       4(AP) - address of current collection buffer (data portion)
                                ;
                                ; IMPLICIT INPUTS:
                                ;
                                ;       None
                                ;
                                ; OUTPUTS:
                                ;
                                ;       None
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;
                                ;       Collection buffer filled with data for each disk.
                                ;
                                ; ROUTINE VALUE:
                                ;
                                ;       R0 = status from SCANJDEVICES routine
                                ;
                                ;       R1 = YES, if subsequent FETCH collection is required.
                                ;       R1 = NO,  if subsequent FETCH collection is NOT required.
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       none
                                ;--
                                ;
                        0000    .ENTRY  JDEVICE_PRE, ^M<>
                                ;
                                        $CMKRNL_S B^SCANJDEVICES,(AP)   ; Scan all jdevice structs in kernel mode
                                ;
                                ; Indicate to caller that FETCH collection is NOT required.
                                ;
                                ;
  51    00000000'8F   D0      1707      MOVL    #NO,R1                  ; FETCH collection NOT required
                      04      1708      RET                             ; Return with status from SCANJDEVICES
```

```
                                                K 4
0675  1710 ;++
0675  1711 ; SCANJDEVICES - subroutine to jdevice data structures in kernel mode
0675  1712 ;
0675  1713 ; CALLING SEQUENCE:
0675  1714 ;
0675  1715 ;       $CMKRNL_S SCANJDEVICES,(AP)
0675  1716 ;
0675  1717 ; INPUTS:
0675  1718 ;
0675  1719 ;       4(AP) - address of current collection buffer (data portion)
0675  1720 ;
0675  1721 ; OUTPUTS:
0675  1722 ;
0675  1723 ;       None
0675  1724 ;
0675  1725 ; IMPLICIT INPUTS:
0675  1726 ;
0675  1727 ;       SCH$LOCKR, SCH$UNLOCK - I/O Mutex lock and unlock routines.
0675  1728 ;       IOC$SCAN_IODB         - Routine which scans the I/O data base
0675  1729 ;                               for the next device/unit.
0675  1730 ;       SCH$GL_CURPCB         - Current PCB.
0675  1731 ;
0675  1732 ; IMPLICIT OUTPUTS:
0675  1733 ;
0675  1734 ;       Collection buffer filled with data for each journal device.
0675  1735 ;
0675  1736 ; ROUTINE VALUE:
0675  1737 ;
0675  1738 ;       R0 = SS$_NORMAL, or system service error status
0675  1739 ;
0675  1740 ; SIDE EFFECTS:
0675  1741 ;
0675  1742 ;       This routine holds the IO MUTEX and runs at ASTDEL IPL while
0675  1743 ;       it is scanning the device data base. When scanning the various
0675  1744 ;       journal device IRP queues, IPL is raised to FORK and lowered
0675  1745 ;       for each queue.
0675  1746 ;--
0675  1747
0675  1748 SCANJDEVICES:
                OFFC  0675  1749         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Register save mask
0677  1750
0677  1751 ;
0677  1752 ; Lock a code segment of this routine in the working set
0677  1753 ; in anticipation of elevating to fork IPL.
0677  1754 ;
0677  1755
0677  1756         ALLOC   8,R0,R3                 ; Get longword pair for $LKWSET
   63   00000703'EF   DE  0684  1757         MOVAL   5$,(R3)                 ; Load addr of first byte to be locked
04 A3   00000775'EF   DE  0688  1758         MOVAL   11$,4(R3)               ; ... and last byte
0693  1759         $LKWSET_S  INADR=(R3)            ; Lock code into working set
          03 50   E8  06A0  1760         BLBS    R0,5$                   ; Continue if OK
           0103   31  06A3  1761         BRW     21$                     ; Else go exit if error
06A6  1762 5$:
   59   04 AC   08   C1  06A6  1763         ADDL3   #MNR_HOM$K_PSIZE,4(AP),R9 ; Point past the prefix to ...
06AB  1764                                      ; ... beginning of data blocks
          58   D4  06AB  1765         CLRL    R8                      ; Clear jdevice counter
   54   00000000'GF   D0  06AD  1766         MOVL    G^SCH$GL_CURPCB,R4      ; Get PCB for IOLOCKR call
```

```
                                06B4  1767          JSB     G^SCH$IOLOCKR                   ; Get mutex to lock I/O data base
                                06BA  1768                                                  ; NOTE -- now at IPL ASTDEL, so can
                                06BA  1769                                                  ; ... take page faults
                                06BA  1770 ;
                                06BA  1771 ; Call IOC$SCAN_IODB to get the next unit in the I/O data base.
                                06BA  1772 ; The unit is described by the DDB and UCB pointers in R11 and
                                06BA  1773 ; R10, respectively. To begin the scan, call SCAN_IODB with R11
                                06BA  1774 ; and R10 containing zero. It returns the first unit in the data
                                06BA  1775 ; base in the same registers. On subsequent calls, simply leave
                                06BA  1776 ; R11 and R10 alone, and SCAN_IODB will return the next unit.
                                06BA  1777 ; If an entire DDB is undesireable, clear R10 before calling
                                06BA  1778 ; and all units for that device will be skipped.
                                06BA  1779 ;
                                06BA  1780
                          5B    D4  06BA  1781          CLRL    R11                     ; Indicate starting at beginning
                          5A    D4  06BC  1782          CLRL    R10                     ; ... of I/O data base
                00000000'GF     16  06BE  1783 10$:     JSB     G^IOC$SCAN_IODB         ; Get the next unit
                       03 50    E8  06C4  1784          BLBS    R0,20$                  ; Branch if we got a unit
                       00B8     31  06C7  1785          BRW     200$                    ; Branch if at end of data base
                                06CA  1786
                                06CA  1787 ;
                                06CA  1788 ; Check the class of the device/unit just provided to see if we want it.
                                06CA  1789 ;
                                06CA  1790
                                06CA  1791 ;
                                06CA  1792 ; Check entire controller (DDB) for jdevice class by examining the UCB.
                                06CA  1793 ; If the DDB class is not jdevice, then clear R10 and branch back to get next
                                06CA  1794 ; device/unit. If it is jdevice, simply continue.
                                06CA  1795 ;
                                06CA  1796
                    A1  8F  91  06CA  1797 20$:     CMPB    #DC$_JOURNAL,-           ; Is the unit a journal device?
                    40  AA      06CD  1798                  UCB$B_DEVCLASS(R10)     ;
                        04  13  06CF  1799          BEQL    30$                     ; Yes, check if it is a template UCB
                        5A  D4  06D1  1800          CLRL    R10                     ; No, skip entire controller
                        E9  11  06D3  1801          BRB     10$                     ; Get first unit on next controller
                                06D5  1802
                                06D5  1803 ;
                                06D5  1804 ; Check if this is a template UCB (templates are always unit 0).
                                06D5  1805 ; Template UCBs will not be displayed since they are only used for
                                06D5  1806 ; cloning purposes and contain no useful information.
                                06D5  1807 ;
                                06D5  1808
                    54  AA  B5  06D5  1809 30$:     TSTW    UCB$W_UNIT(R10)         ; Is this a template UCB?
                        E4  13  06D8  1810          BEQL    10$                     ; Yes, get next UCB
                                06DA  1811                                          ; No, treat it as a normal UCB
                                06DA  1812
                                06DA  1813 ;
                                06DA  1814 ; R11/R10 now point to a relevant journal DDB/UCB.  Collect pertinent data.
                                06DA  1815 ;
                                06DA  1816
                    50  34  AB  D0  06DA  1817          MOVL    DDB$L_SB(R11),R0        ; Get system block pointer
                            0F  13  06DE  1818          BEQL    40$                     ; Br if none
                00000000'8F     50  D1  06E0  1819          CMPL    R0,#SCS$GA_LOCALSB      ; Disk on the local system?
                            06  13  06E7  1820          BEQL    40$                     ; Yes -- skip node name
                    89  44  A0  7D  06E9  1821          MOVQ    SB$T_NODENAME(R0),(R9)+ ; Collect the node name
                            02  11  06ED  1822          BRB     50$                     ; Get device name
                                06EF  1823 40$:
```

```
                89   7C  06EF   1824                CLRQ    (R9)+                       ; Null node name
                                06F1   1825 50$:
          89    14  AB  D0  06F1   1826            MOVL    DDB$T_NAME(R11),(R9)+       ; Collect the device name
          89    54  AA  B0  06F5   1827            MOVW    UCB$W_UNIT(R10),(R9)+       ; Collect the (binary) unit number
              00E8  CA  D0  06F9   1828            MOVL    UCB$L_JNL_WRCNT(R10),-      ; Collect the journal write count
                89      06FD   1829                        (R9)+
              00EC  CA  D0  06FE   1830            MOVL    UCB$L_JNL_BWCNT(R10),-      ; Collect the journal buffer
                89      0702   1831                        (R9)+                       ;  write count
```

```
                     0703  1833 55$:                                ; Beginning of locked section
50   0B AA   9A      0703  1834          MOVZBL  UCB$B_FIPL(R10),R0  ; Get fork IPL
                     0707  1835
                     0707  1836 ;
                     0707  1837 ; Sum the number of entries in the journal UCB's normal queue into R7.
                     0707  1838 ;
                     0707  1839
        57   D4      0707  1840          CLRL    R7                  ; Clear queue entry counter
                     0709  1841          DSBINT  R0                  ; Elevate to fork IPL to access IRPs
     55 6A   DE      070F  1842          MOVAL   UCB$L_FQFL(R10),R5  ; Get address of normal queue header
     55 6A   D1      0712  1843          CMPL    UCB$L_FQFL(R10),R5  ; Is the queue empty?
        0D   13      0715  1844          BEQL    70$                 ; Yes, go store count
     56 55   D0      0717  1845          MOVL    R5,R6               ; No, copy the queue header
        57   D6      071A  1846 60$:     INCL    R7                  ; Count this as a queue entry
     56 66   D0      071C  1847          MOVL    IRP$L_IOQFL(R6),R6  ; Point to next possible entry
     55 66   D1      071F  1848          CMPL    IRP$L_IOQFL(R6),R5  ; Is there another entry?
        F6   12      0722  1849          BNEQ    60$                 ; Yes, go look for another entry
                     0724  1850                                      ; No, we're done
                     0724  1851 70$:     ENBINT                      ; Back to IPL$_ASTDEL for coll buff ref
     89 57   D0      0727  1852          MOVL    R7,(R9)+            ; Collect the sum of the queue entries
                     072A  1853
                     072A  1854 ;
                     072A  1855 ; Sum the number of entries in the journal UCB's wait queue into R7.
                     072A  1856 ;
                     072A  1857
        57   D4      072A  1858          CLRL    R7                  ; Clear queue entry counter
                     072C  1859          DSBINT  R0                  ; Elevate to fork IPL to access IRPs
55   00A8 CA   DE      0732  1860          MOVAL   UCB$L_JNL_WQFL(R10),R5 ; Get address of wait queue header
55   00A8 CA   D1      0737  1861          CMPL    UCB$L_JNL_WQFL(R10),R5 ; Is the queue empty?
        0D   13      073C  1862          BEQL    90$                 ; Yes, go store count
     56 55   D0      073E  1863          MOVL    R5,R6               ; No, copy the queue header
        57   D6      0741  1864 80$:     INCL    R7                  ; Count this as a queue entry
     56 66   D0      0743  1865          MOVL    IRP$L_IOQFL(R6),R6  ; Point to next possible entry
     55 66   D1      0746  1866          CMPL    IRP$L_IOQFL(R6),R5  ; Is there another entry?
        F6   12      0749  1867          BNEQ    80$                 ; Yes, go look for another entry
                     074B  1868                                      ; No, we're done
                     074B  1869 90$:     ENBINT                      ; Back to IPL$_ASTDEL for coll buff ref
     89 57   D0      074E  1870          MOVL    R7,(R9)+            ; Collect the sum of the queue entries
                     0751  1871
                     0751  1872 ;
                     0751  1873 ; Sum the number of entries in the journal UCB's force queue into R7.
                     0751  1874 ;
                     0751  1875
        57   D4      0751  1876          CLRL    R7                  ; Clear queue entry counter
                     0753  1877          DSBINT  R0                  ; Elevate to fork IPL to access IRPs
55   00B0 CA   DE      0759  1878          MOVAL   UCB$L_JNL_FQFL(R10),R5 ; Get address of force queue header
55   00B0 CA   D1      075E  1879          CMPL    UCB$L_JNL_FQFL(R10),R5 ; Is the queue empty?
        0D   13      0763  1880          BEQL    110$                ; Yes, go store count
     56 55   D0      0765  1881          MOVL    R5,R6               ; No, copy the queue header
        57   D6      0768  1882 100$:    INCL    R7                  ; Count this as a queue entry
     56 66   D0      076A  1883          MOVL    IRP$L_IOQFL(R6),R6  ; Point to next possible entry
     55 66   D1      076D  1884          CMPL    IRP$L_IOQFL(R6),R5  ; Is there another entry?
        F6   12      0770  1885          BNEQ    100$                ; Yes, go look for another entry
                     0772  1886                                      ; No, we're done
                     0772  1887 110$:    ENBINT                      ; Return to IPL$_ASTDEL
                     0775  1888 115$:                                ; End of locked section
                     0775  1889
```

```
                89    57   D0  0775  1890          MOVL    R7,(R9)+                        ; Collect the sum of the queue entries
            00F0  CA   D0  0778  1891          MOVL    UCB$L_JNL_EXCNT(R10),-          ; Collect the extend rate
                89         077C  1892                  (R9)+                           ;
                58   D6  077D  1893          INCL    R8                              ; Count this unit
              FF3C   31  077F  1894          BRW     10$                             ; Go get next device/unit
```

```
                                                C 5
                     0782  1896 ;
                     0782  1897 ; The entire I/O data base has been scanned.  Relinquish the I/O Mutex
                     0782  1898 ; and drop IPL back to 0.
                     0782  1899 ;
                     0782  1900
54  00000000'GF   DO 0782  1901 200$:   MOVL    G^SCH$GL_CURPCB,R4          ; Get PCB for IOUNLOCK call
    00000000'GF   16 0789  1902          JSB     G^SCH$IOUNLOCK              ; Relinquish lock on I/O data base
                     078F  1903                                              ; NOTE -- this rtn clobbers R0-R2
                     078F  1904          SETIPL  #0                          ; Return to IPL 0
                     0792  1905
     50  04 AC   DO 0792  1906          MOVL    4(AP),R0                    ; Point to prefix part of coll buff
        60  58   DO 0796  1907          MOVL    R8,MNR_HOM$L_ELTCT(R0)      ; Save element count
         04 A0   D4 0799  1908          CLRL    MNR_HOM$L_RESERVED(R0)      ; Clear reserved longword
                     079C  1909          $ULWSET_S  INADR=(R3)               ; Unlock code from working set
                     07A9  1910 210$:
                  04 07A9  1911          RET                                 ; Return with status
                     07AA  1912
                     07AA  1913
```

```
                        07AA  1915          .SBTTL  SCS_PRE - SCS Class Pre-collection Rtn
                        07AA  1916 ;++
                        07AA  1917 ;
                        07AA  1918 ; FUNCTIONAL DESCRIPTION:
                        07AA  1919 ;
                        07AA  1920 ;       Loop through SCS data base, collecting info on each node.
                        07AA  1921 ;       The info is stored in the collection buffer passed to this
                        07AA  1922 ;       rtn by the FETCH rtn.  System blocks for UDAs are discarded.
                        07AA  1923 ;
                        07AA  1924 ; CALLING SEQUENCE:
                        07AA  1925 ;
                        07AA  1926 ;       CALLS/CALLG
                        07AA  1927 ;
                        07AA  1928 ; INPUTS:
                        07AA  1929 ;
                        07AA  1930 ;       4(AP) - address of current collection buffer (data portion)
                        07AA  1931 ;
                        07AA  1932 ; IMPLICIT INPUTS:
                        07AA  1933 ;
                        07AA  1934 ;       None
                        07AA  1935 ;
                        07AA  1936 ; OUTPUTS:
                        07AA  1937 ;
                        07AA  1938 ;       None
                        07AA  1939 ;
                        07AA  1940 ; IMPLICIT OUTPUTS:
                        07AA  1941 ;
                        07AA  1942 ;       Collection buffer filled with data for each node.
                        07AA  1943 ;
                        07AA  1944 ; ROUTINE VALUE:
                        07AA  1945 ;
                        07AA  1946 ;       R0 = status from SCANSCS routine
                        07AA  1947 ;       R1 = NO, since subsequent FETCH collection is NOT required.
                        07AA  1948 ;
                        07AA  1949 ; SIDE EFFECTS:
                        07AA  1950 ;
                        07AA  1951 ;       none
                        07AA  1952 ;--
                        07AA  1953
                0000    07AA  1954 .ENTRY  SCS_PRE, ^M<>
                        07AC  1955
                        07AC  1956          $CMKRNL_S B^SCANSCS,(AP)        ; Scan all SCS structs in kernel mode
                        07B8  1957
                        07B8  1958 ;
                        07B8  1959 ; Indicate to caller that FETCH collection is NOT required.
                        07B8  1960 ;
                        07B8  1961
     51 00000000'8F     D0  07B8  1962          MOVL    #NO,R1                  ; FETCH collection NOT required
                        04  07BF  1963          RET                             ; Return
```

```
                        07C0  1965 ;++
                        07C0  1966 ; SCANSCS - subroutine to SCS data structures in kernel mode
                        07C0  1967 ;
                        07C0  1968 ; CALLING SEQUENCE:
                        07C0  1969 ;
                        07C0  1970 ;       $CMKRNL_S SCANSCS,(AP)
                        07C0  1971 ;
                        07C0  1972 ; INPUTS:
                        07C0  1973 ;
                        07C0  1974 ;       4(AP) - address of current collection buffer (data portion)
                        07C0  1975 ;
                        07C0  1976 ; OUTPUTS:
                        07C0  1977 ;
                        07C0  1978 ;       None
                        07C0  1979 ;
                        07C0  1980 ; IMPLICIT INPUTS:
                        07C0  1981 ;
                        07C0  1982 ;       None
                        07C0  1983 ;
                        07C0  1984 ; IMPLICIT OUTPUTS:
                        07C0  1985 ;
                        07C0  1986 ;       Collection buffer filled with data for each node.
                        07C0  1987 ;
                        07C0  1988 ; SIDE EFFECTS:
                        07C0  1989 ;
                        07C0  1990 ;       This routine runs at SCS IPL while it is scanning the SCS data base.
                        07C0  1991 ;--
                        07C0  1992
                        07C0  1993 SCANSCS:
                   0FFC 07C0  1994         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Register save mask
                        07C2  1995
                        07C2  1996 ;
                        07C2  1997 ; Lock the  entire collection buffer down, point R9 to the data portion of
                        07C2  1998 ; the collection buffer, and clear the node counter (R8).  If there are
                        07C2  1999 ; few nodes, locking down the entire collection buffer may not be necessary.
                        07C2  2000 ;
                        07C2  2001
                        07C2  2002         ALLOC   8,R0,R11                ; Get longword pair for $LKWSET
5A  00000000'EF   D0    07CF  2003         MOVL    CDBPTR,R10              ; Get SCS class pointer
 5A    20 AA      3C    07D6  2004         MOVZWL  CDB$W_BLKLEN(R10),R10   ; Calculate the ending address of
5A  00000000'8F   C4    07DA  2005         MULL2   #MAXECTS,R10            ;  the entire homogenous buffer
     5A    08     C0    07E1  2006         ADDL2   #MNR_HOM$K_PSIZE,R10    ;  to be used in the second
   5A    04 AC   C0    07E4  2007         ADDL2   4(AP),R10               ;  longword of the $LKWSET pair
   6B    04 AC   D0    07E8  2008         MOVL    4(AP),(R11)             ; Load addr of first byte to be locked
   04 AB   5A    D0    07EC  2009         MOVL    R10,4(R11)              ; ... and last byte
                       07F0  2010         $LKWSET_S  INADR=(R11)          ; Lock collection buffer into Wkset
         03 50   E8    07F3  2011         BLBS    R0,10$                  ; Continue if OK
          00FE   31    07F6  2012         BRW     25$                     ; Else go exit if error
            08   C1    07F9  2013 10$:    ADDL3   #MNR_HOM$K_PSIZE,-      ; Point past the prefix to ...
   59    04 AC         07FB  2014                 4(AP),R9                ; ... beginning of data blocks
               58   D4 07FE  2015         CLRL    R8                      ; Clear SCS node counter
                       0800  2016
```

```
                        080A  2018 ;++
                        080A  2019 ; The collection buffer has been locked down, now sum all the counters in the
                        080A  2020 ; CDT's for a given node (non-UDA system block) into the collection buffer.
                        080A  2021 ;
                        080A  2022 ; SCS$GQ_CONFIG                          SB:  System Block
                        080A  2023 ;    |                                   PB:  Path Block
                        080A  2024 ;    ----------> +------+                CDT: Connection Descriptor Table
                        080A  2025 ;                | SB   |
                        080A  2026 ;                |head  |
                        080A  2027 ;                +------+
                        080A  2028 ;                  ||
                        080A  2029 ;                +------+     +------+          +------+
                        080A  2030 ;                | SB   |=====| PB   |=====  =====| PB   |
                        080A  2031 ;                |      |     |      |          |      |
                        080A  2032 ;                +------+     +------+          +------+
                        080A  2033 ;                  ||                             |
                        080A  2034 ;                  ||                           +------+
                        080A  2035 ;                  ||                           | CDT  |
                        080A  2036 ;                                               |      |
                        080A  2037 ;                  ||                           +------+
                        080A  2038 ;                  ||                             |
                        080A  2039 ;                  ||                             |
                        080A  2040 ;                  ||                             |
                        080A  2041 ;                +------+                       +------+
                        080A  2042 ;                | SB   |                       | CDT  |
                        080A  2043 ;                |      |                       |      |
                        080A  2044 ;                +------+                       +------+
                        080A  2045 ;--
                        080A  2046
                        080A  2047 20$:     DSBINT  300$                        ; Raise to SCS IPL
          5A  00000000'GF   DE  0814  2048          MOVAL   G^SCS$GQ_CONFIG,R10     ; Get the address of the system blk hdr
                    5A  6A  D0  081B  2049 30$:     MOVL    SB$L_FLINK(R10),R10     ; Get the next system block
          00000000'8F   5A  D1  081E  2050          CMPL    R10,#SCS$GQ_CONFIG      ; All system blocks done?
                      03  12  0825  2051          BNEQ    40$                     ; No, check if it is a UDA system block
                    008D  31  0827  2052          BRW     200$                    ; Yes, finish up call
                   24 AA  95  082A  2053 40$:     TSTB    SB$T_SWTYPE(R10)        ; Is it a UDA?
                      EC  13  082D  2054          BEQL    30$                     ; Yes, get next system block
                          082F  2055
                          082F  2056 ;
                          082F  2057 ; The system block was for a remote node.  Clear all of the SCS counters
                          082F  2058 ;
                          082F  2059
08 A9   3C   00   08 A9   00  2C  082F  2060 50$:     MOVC5   #00, -                  ; Zero out the data area for this
                          0837  2061                  MNR_SCS$L_DGSENT(R9), -     ;  node in the collection buffer
                          0837  2062                  #00, -                      ;
                          0837  2063                  #<MNR_SCS$C_CBWORK-MNR_SCS$L_DGSENT>, - ;
                          0837  2064                  MNR_SCS$L_DGSENT(R9)        ;
                    55  0C AA  DE  0837  2065          MOVAL   SB$L_PBFL(R10),R5       ; Save address of path block listhead
                    56  0C AA  D0  083B  2066          MOVL    SB$L_PBFL(R10),R6       ; Get the address of 1st path block
                 55   56      D1  083F  2067 60$:     CMPL    R6,R5                   ; Any more path blocks?
                         73   13  0842  2068          BEQL    110$                    ; No, get next system block
                    57  34 A6  D0  0844  2069 70$:     MOVL    PB$L_CDTLST(R6),R7      ; Yes, get 1st connection desc. table
                         68   13  0848  2070          BEQL    100$                    ; If no more CDTs, get next path block
                          084A  2071
                          084A  2072 ;
                          084A  2073 ; Sum the values from this connection descriptor table into the collection
                          084A  2074 ; buffer for this system block.
```

```
                                084A  2075 ;
                                084A  2076
                     70 A7   C0 084A  2077 80$:    ADDL2   CDT$L_DGSENT(R7),-      ; Sum # application DGs sent
                     08 A9      084D  2078          MNR_SCS$L_DGSENT(R9)
                                084F  2079
                     74 A7   C0 084F  2080          ADDL2   CDT$L_DGRCVD(R7),-      ; Sum # application DGs received
                     0C A9      0852  2081          MNR_SCS$L_DGRCVD(R9)
                                0854  2082
                     78 A7   C0 0854  2083          ADDL2   CDT$L_DGDISCARD(R7),-   ; Sum # application DGs discarded
                     10 A9      0857  2084          MNR_SCS$L_DGDISCARD(R9)
                                0859  2085
                     7C A7   C0 0859  2086          ADDL2   CDT$L_MSGSENT(R7),-     ; Sum # application msgs sent
                     14 A9      085C  2087          MNR_SCS$L_MSGSENT(R9)
                                085E  2088
                   0080 C7   C0 085E  2089          ADDL2   CDT$L_MSGRCVD(R7),-     ; Sum # application msgs received
                     18 A9      0862  2090          MNR_SCS$L_MSGRCVD(R9)
                                0864  2091
                   0084 C7   C0 0864  2092          ADDL2   CDT$L_SNDDATS(R7),-     ; Sum # block send datas initiated
                     1C A9      0868  2093          MNR_SCS$L_SNDDATS(R9)
                                086A  2094
                   0088 C7   C0 086A  2095          ADDL2   CDT$L_BYTSENT(R7),-     ; Sum # bytes sent via send datas
                     20 A9      086E  2096          MNR_SCS$L_KBYTSENT(R9)
                        08   1E 0870  2097          BCC     82$                     ; Byte count overflow longword?
               00800000 8F   C0 0872  2098          ADDL2   #^X00800000,-           ; Yes, update Kbyte counter
                     38 A9      0878  2099          MNR_SCS$L_CBKBSENT(R9)
                                087A  2100
                   008C C7   C0 087A  2101 82$:    ADDL2   CDT$L_REQDATS(R7),-     ; Sum # block request datas initiated
                     24 A9      087E  2102          MNR_SCS$L_REQDATS(R9)
                                0880  2103
                   0090 C7   C0 0880  2104          ADDL2   CDT$L_BYTREQD(R7),-     ; Sum # bytes received via req datas
                     28 A9      0884  2105          MNR_SCS$L_KBYTREQD(R9)
                        08   1E 0886  2106          BCC     84$                     ; Byte count overflow longword?
               00800000 8F   C0 0888  2107          ADDL2   #^X00800000,-           ; Yes, update Kbyte counter
                     3C A9      088E  2108          MNR_SCS$L_CBKBREQD(R9)
                                0890  2109
                   0094 C7   C0 0890  2110 84$:    ADDL2   CDT$L_BYTMAPD(R7),-     ; Sum # bytes mapped for block xfr
                     2C A9      0894  2111          MNR_SCS$L_KBYTMAPD(R9)
                        08   1E 0896  2112          BCC     86$                     ; Byte count overflow longword?
               00800000 8F   C0 0898  2113          ADDL2   #^X00800000,-           ; Yes, update Kbyte counter
                     40 A9      089E  2114          MNR_SCS$L_CBKBMAPD(R9)
                                08A0  2115
                   0098 C7   A0 08A0  2116 86$:    ADDW2   CDT$W_QCR_CNT(R7),-     ; Sum # times conn. q'd for send credit
                     30 A9      08A4  2117          MNR_SCS$L_QCR_CNT(R9)
                                08A6  2118
                   009A C7   A0 08A6  2119          ADDW2   CDT$W_QBDT_CNT(R7),-    ; Sum # times conn. q'd for buff descr
                     34 A9      08AA  2120          MNR_SCS$L_QBDT_CNT(R9)
                                08AC  2121
                57   6C A7   D0 08AC  2122 90$:    MOVL    CDT$L_CDTLST(R7),R7     ; Get the next connection desc. table
                        98   12 08B0  2123          BNEQ    80$                     ; If another CDT, sum the CDT's counters
                                08B2  2124
                                08B2  2125 ;
                                08B2  2126 ; All the CDTs have been summed for this path block.  Get the next path block.
                                08B2  2127 ;
                                08B2  2128
                56   66   D0 08B2  2129 100$:   MOVL    PB$L_FLINK(R6),R6       ; No more, get next path block
                        88   11 08B5  2130          BRB     60$                     ; Check if all path blocks done
                                08B7  2131
```

```
                                                H 5
                     08B7  2132 ;
                     08B7  2133 ; There are no more path blocks for this system block, thus no more CDTs.
                     08B7  2134 ; The counters were summed into the collection buffer, so just the node
                     08B7  2135 ; name is left to be placed in the collection buffer.  The byte counts
                     08B7  2136 ; that were stored in the collection buffer are converted to Kbytes.
                     08B7  2137 ;
                     08B7  2138
        58   D6   08B7  2139 110$:   INCL    R8                              ; Increment node (system block) counter
    44 AA    7D   08B9  2140         MOVQ    SB$T_NODENAME(R10),-            ; Collect the node name
        69                08BC  2141                 MNR_SCS$Q_NODENAME(R9)  ;  for this system block
    16   0A   EF   08BD  2142         EXTZV   #10,#22,-                       ; Convert # bytes sent via send datas
      20 A9             08C0  2143                 MNR_SCS$L_KBYTSENT(R9),-;  to Kbytes
      20 A9             08C2  2144                 MNR_SCS$L_KBYTSENT(R9)  ;
      38 A9   C0   08C4  2145         ADDL2   MNR_SCS$L_CBKBSENT(R9),-;  Add in any Kbytes from
      20 A9             08C7  2146                 MNR_SCS$L_KBYTSENT(R9)  ;  BYTSENT longword overflow
    16   0A   EF   08C9  2147         EXTZV   #10,#22,-                       ; Convert # bytes sent via request
      28 A9             08CC  2148                 MNR_SCS$L_KBYTREQD(R9),-;  datas to Kbytes
      28 A9             08CE  2149                 MNR_SCS$L_KBYTREQD(R9)  ;
      3C A9   C0   08D0  2150         ADDL2   MNR_SCS$L_CBKBREQD(R9),-;  Add in any Kbytes from
      28 A9             08D3  2151                 MNR_SCS$L_KBYTREQD(R9)  ;  BYTREQD longword overflow
    16   0A   EF   08D5  2152         EXTZV   #10,#22,-                       ; Convert # bytes sent via mapped
      2C A9             08D8  2153                 MNR_SCS$L_KBYTMAPD(R9),-;  transfer to Kbytes
      2C A9             08DA  2154                 MNR_SCS$L_KBYTMAPD(R9)  ;
      40 A9   C0   08DC  2155         ADDL2   MNR_SCS$L_CBKBMAPD(R9),-;  Add in any Kbytes from
      2C A9             08DF  2156                 MNR_SCS$L_KBYTMAPD(R9)  ;  BYTMAPD longword overflow
    59   38   C0   08E1  2157         ADDL2   #MNR_SCS$C_CBLENGTH,R9          ; Point to coll. buff. space for next SB
      FF34   31   08E4  2158         BRW     30$                             ; Look for the next system block
                     08E7  2159
                     08E7  2160 ;
                     08E7  2161 ; The entire SCS data base has been scanned.  Drop IPL back to 0,
                     08E7  2162 ; unlock the collection buffer, and return.
                     08E7  2163 ;
                     08E7  2164
                     08E7  2165 200$:   ENBINT                               ; Back to IPL 0
50    04 AC    D0   08EA  2166         MOVL    4(AP),R0                        ; Point to prefix part of coll buff
   60    58    D0   08EE  2167         MOVL    R8,MNR_HOM$L_ELTCT(R0)          ; Save element count
      04 A0    D4   08F1  2168         CLRL    MNR_HOM$L_RESERVED(R0)          ; Clear reserved longword
                     08F4  2169         $ULWSET_S  INADR=(R11)                  ; Unlock code from working set
              04   0901  2170 250$:   RET                                     ; Return
                     0902  2171
        00000008     0902  2172 300$:   .LONG   IPL$_SCS
                     0906  2173         ASSUME  .-20$  LE  512                  ; Make sure it doesn't exceed two pages
                     0906  2174
```

```
                                                                0906  2176          .SBTTL  FSCACHE_PRE - File System Cache Pre-collection Rtn
                                                                0906  2177 ;++
                                                                0906  2178 ;
                                                                0906  2179 ; FUNCTIONAL DESCRIPTION:
                                                                0906  2180 ;
                                                                0906  2181 ;       Store the total of hits + misses in the appropriate global locations
                                                                0906  2182 ;       for each file system cache, later to be moved into the collection buffer
                                                                0906  2183 ;       by the FETCH routine.
                                                                0906  2184 ;
                                                                0906  2185 ; CALLING SEQUENCE:
                                                                0906  2186 ;
                                                                0906  2187 ;       CALLS/CALLG
                                                                0906  2188 ;
                                                                0906  2189 ; INPUTS:
                                                                0906  2190 ;
                                                                0906  2191 ;       4(AP) - address of current collection buffer (data portion)
                                                                0906  2192 ;
                                                                0906  2193 ; IMPLICIT INPUTS:
                                                                0906  2194 ;
                                                                0906  2195 ;       None
                                                                0906  2196 ;
                                                                0906  2197 ; OUTPUTS:
                                                                0906  2198 ;
                                                                0906  2199 ;       None
                                                                0906  2200 ;
                                                                0906  2201 ; IMPLICIT OUTPUTS:
                                                                0906  2202 ;
                                                                0906  2203 ;       Global locations filled with (hits + misses) for each cache.
                                                                0906  2204 ;
                                                                0906  2205 ; ROUTINE VALUE:
                                                                0906  2206 ;
                                                                0906  2207 ;       R0 = SS$_NORMAL
                                                                0906  2208 ;
                                                                0906  2209 ;       R1 = YES, if subsequent FETCH collection is required.
                                                                0906  2210 ;       R1 = NO,  if subsequent FETCH collection is NOT required.
                                                                0906  2211 ;
                                                                0906  2212 ; SIDE EFFECTS:
                                                                0906  2213 ;
                                                                0906  2214 ;       none
                                                                0906  2215 ;--
                                                        0000    0906  2216 .ENTRY  FSCACHE_PRE, ^M<>
                                                                0908  2217
00000000'EF   00000000'EF   C1  0908  2218          ADDL3   PMS$GL_FILHDR_HIT,PMS$GL_FILHDR_MISS,-
              000000A0'EF       0913  2219                  FILHDR_TRIES                ;save sum of hits + misses
00000000'EF   00000000'EF   C1  0918  2220          ADDL3   PMS$GL_FIDHIT,PMS$GL_FIDMISS,-
              000000A4'EF       0923  2221                  FID_TRIES                   ;save sum of hits + misses
00000000'EF   00000000'EF   C1  0928  2222          ADDL3   PMS$GL_DIRHIT,PMS$GL_DIRMISS,-
              000000A8'EF       0933  2223                  DIRFCB_TRIES                ;save sum of hits + misses
00000000'EF   00000000'EF   C1  0938  2224          ADDL3   PMS$GL_DIRDATA_HIT,PMS$GL_DIRDATA_MISS,-
              000000AC'EF       0943  2225                  DIRDATA_TRIES               ;save sum of hits + misses
00000000'EF   00000000'EF   C1  0948  2226          ADDL3   PMS$GL_EXTHIT,PMS$GL_EXTMISS,-
              00000080'EF       0953  2227                  EXT_TRIES                   ;save sum of hits + misses
00000000'EF   00000000'EF   C1  0958  2228          ADDL3   PMS$GL_QUOHIT,PMS$GL_QUOMISS,-
              000000B4'EF       0963  2229                  QUO_TRIES                   ;save sum of hits + misses
00000000'EF   00000000'EF   C1  0968  2230          ADDL3   PMS$GL_STORAGMAP_HIT,PMS$GL_STORAGMAP_MISS,-
              00000088'EF       0973  2231                  STORAGMAP_TRIES             ;save sum of hits + misses
                                0978  2232 ;
```

```
                     0978  2233 ; Indicate to caller that FETCH collection IS required.
                     0978  2234 ;
51   00000000'8F  D0 0978  2235         MOVL    #YES,R1                 ; FETCH collection required
50   00000000'8F  D0 097F  2236         MOVL    #SS$_NORMAL,R0          ; success status
                  04 0986  2237         RET                             ; return
                     0987  2238 .END
```

```
ALL_STAT                =  00000000
AVE_STAT                =  00000002
BARSIZE                    ********    X   03
BASE                       00000074 R      01
BIGHOLE                    00000020 RG     01
BLANKS                     000000BC R      01
BLKAST                     00000058 RG     01
CDB                     =  00000000
CDB$A_BUFFERS           =  0000002E
CDB$A_CDX               =  00000032
CDB$A_CMDHDR            =  0000004F
CDB$A_FAOCTR            =  00000004
CDB$A_ITMSTR            =  0000001C
CDB$A_POSTCOLL          =  00000026
CDB$A_PRECOLL           =  00000022
CDB$A_SUMBUF            =  0000000C
CDB$A_TITLE             =  00000010
CDB$B_FAOPRELEN         =  00000041
CDB$B_FAOSEGLEN         =  00000040
CDB$B_ST                =  00000042
CDB$B_ST_CUR            =  00000044
CDB$B_ST_DEF            =  00000043
CDB$K_SIZE              =  00000053
CDB$L_BUFFERS           =  0000002A
CDB$L_ECOUNT            =  00000018
CDB$L_FAOCTR            =  00000000
CDB$L_FLAGS             =  0000004B
CDB$L_ICOUNT            =  00000014
CDB$L_MIN               =  00000038
CDB$L_RANGE             =  0000003C
CDB$L_SUMBUF            =  00000008
CDB$M_CPU               =  00000002
CDB$M_CPU_COMB          =  00000008
CDB$M_CTPRES            =  00000001
CDB$M_DISABLE           =  00000200
CDB$M_DISKAC            =  00000040
CDB$M_DISKVN            =  00000080
CDB$M_EXPLIC            =  00001000
CDB$M_HOMOG             =  00000020
CDB$M_KUNITS            =  00000400
CDB$M_PERCENT           =  00000001
CDB$M_STD               =  00000010
CDB$M_SWAPBUF           =  00000002
CDB$M_SYSCLS            =  00000100
CDB$M_UNIFORM           =  00000004
CDB$M_WIDE              =  00000800
CDB$S_CDB               =  00000053
CDB$S_FILLER            =  00000013
CDB$S_FLAGS             =  00000004
CDB$S_QFILLER           =  0000000E
CDB$S_QFLAGS            =  00000002
CDB$V_CPU               =  00000001
CDB$V_CPU_COMB          =  00000003
CDB$V_CTPRES            =  00000000
CDB$V_DISABLE           =  00000009
CDB$V_DISKAC            =  00000006
CDB$V_DISKVN            =  00000007
CDB$V_EXPLIC            =  0000000C
CDB$V_FILLER            =  0000000D
CDB$V_HOMOG             =  00000005
CDB$V_KUNITS            =  0000000A
CDB$V_PERCENT           =  00000000
CDB$V_QFILLER           =  00000002
CDB$V_STD               =  00000004
CDB$V_SWAPBUF           =  00000001
CDB$V_SYSCLS            =  00000008
CDB$V_UNIFORM           =  00000002
CDB$V_WIDE              =  0000000B
CDB$W_BLKLEN            =  00000020
CDB$W_DISPCTL           =  00000036
CDB$W_QFLAGS            =  00000045
CDB$W_QFLAGS_CUR        =  00000049
CDB$W_QFLAGS_DEF        =  00000047
CDBPTR                     ********    X   03
CDT$L_BYTMAPD           =  00000094
CDT$L_BYTREQD           =  00000090
CDT$L_BYTSENT           =  00000088
CDT$L_CDTLST            =  0000006C
CDT$L_DGDISCARD         =  00000078
CDT$L_DGRCVD            =  00000074
CDT$L_DGSENT            =  00000070
CDT$L_MSGRCVD           =  00000080
CDT$L_MSGSENT           =  0000007C
CDT$L_REQDATS           =  0000008C
CDT$L_SNDDATS           =  00000084
CDT$W_QBDT_CNT          =  0000009A
CDT$W_QCR_CNT           =  00000098
CLASS_HDR               =  00000000
COUNT_RES                  00000203 R      03
CPU_BUSY                   00000068 RG     01
CUR_STAT                =  00000001
DC$_DISK                =  00000001
DC$_JOURNAL             =  000000A1
DDB$L_ALLOCLS           =  0000003C
DDB$L_SB                =  00000034
DDB$T_NAME              =  00000014
DECNET_PRE                 0000025F RG     03
DEF$A_DISP              =  0000000C
DEF$A_REC               =  00000004
DEF$A_SUMM              =  00000014
DEF$L_DISP              =  00000008
DEF$L_REC               =  00000000
DEF$L_SUMM              =  00000010
DEF$S_DEF_DESC          =  00000018
DEF_DESC                =  00000000
DEQ                        00000054 RG     01
DEV$V_CDP               =  00000003
DEV$V_MNT               =  00000013
DIRDATA_TRIES              000000AC RG     01
DIRFCB_TRIES               000000A8 RG     01
DISK_PRE                   000005A0 RG     03
DLCKMSGS                   00000064 RG     01
DLOCK_PRE                  0000023E RG     03
DYNINOSE                   00000044 RG     01
```

```
ENQCVT                  00000050 RG    01
ENQNEW                  0000004C RG    01
EXE$GL_MP               ******** X     03
EXE$GL_NONPAGED         ******** X     03
EXT_TRIES               000000B0 RG    01
FCPCACHE                00000004 RG    01
FCPCALLS                00000000 RG    01
FCPCPU                  00000008 RG    01
FCPFAULT                00000014 RG    01
FCPREAD                 0000000C RG    01
FCPWRITE                00000010 RG    01
FCP_PRE                 00000000 RG    03
FID_TRIES               000000A4 RG    01
FILE_HDR              = 00000000
FILHDR_TRIES            000000A0 RG    01
FSCACHE_PRE             00000906 RG    03
GETSEC                  00000417 R     03
HOLECNT                 00000018 RG    01
HOLESUM                 0000001C RG    01
HOM_CLASS_PRE         = 00000000
IOC$GL_IRPCNT           ******** X     03
IOC$GL_IRPFL            ******** X     03
IOC$GL_LRPCNT           ******** X     03
IOC$GL_LRPFL            ******** X     03
IOC$GL_SRPCNT           ******** X     03
IOC$GL_SRPFL            ******** X     03
IOC$SCAN_IODB           ******** X     03
IPL$_SCS              = 00000008
IPL$_SYNCH            = 00000008
IRP$C_IOQFL           = 00000000
IRPCNT                  0000002C RG    01
IRPINUSE                0000003C RG    01
JDEVICE_PRE             0000065F RG    03
LCK$GL_HASHTBL          ******** X     03
LCK$GL_HTBLCNT          ******** X     03
LCK$GL_IDTBL            ******** X     03
LCK$GL_MAXID            ******** X     03
LOCKCNT                 0000005C RG    01
LOCK_PRE                0000016E RG    03
LRPCNT                  00000030 RG    01
LRPINUSE                00000040 RG    01
MAXELTS                 ******** X     03
MAX_STAT              = 00000004
MBP                   = 00000000
MBP$A_ADDR            = 00000018
MBP$A_B1ST            = 00000004
MBP$A_BA              = 00000000
MBP$A_BUFF1ST         = 00000004
MBP$A_BUFFA           = 00000000
MBP$A_BUFFERA         = 00000000
MBP$A_BUFFERB         = 00000004
MBP$A_DATA            = 00000008
MBP$A_DIFF            = 0000000C
MBP$A_MAX             = 00000010
MBP$A_MIN             = 0000000C
MBP$A_ORDER           = 00000010
MBP$A_PCMAX           = 00000020
MBP$A_PCMIN           = 0000001C
MBP$A_PCSTATS         = 00000018
MBP$A_PCSUM           = 00000024
MBP$A_PID             = 00000014
MBP$A_PR_FAOSTK       = 00000008
MBP$A_STATS           = 00000008
MBP$A_SUM             = 00000014
MBP$K_SIZE            = 00000028
MBP$S_MBP             = 00000028
MBP$S_MBP2            = 0000001C
MBP$S_MBP3            = 0000000C
MBP2                  = 00000000
MBP3                  = 00000000
MCA                   = 00000000
MCA$A_INPUT_PTR       = 00000004
MCA$A_MPADDR          = 0000001C
MCA$B_FIRSTC          = 00000030
MCA$B_LASTC           = 00000031
MCA$K_SIZE            = 0000003A
MCA$L_COLLCNT         = 0000000C
MCA$L_CONSEC_REC      = 00000034
MCA$L_DISPCNT         = 00000010
MCA$L_INPUT_LEN       = 00000000
MCA$L_INTTICKS        = 00000008
MCA$L_INT_MULT        = 00000014
MCA$L_PROC_DISP       = 00000018
MCA$Q_CURR_TIME       = 00000020
MCA$Q_LASTCOLL        = 00000028
MCA$S_CURR_TIME       = 00000008
MCA$S_FILLER          = 00000006
MCA$S_FLAGS           = 00000002
MCA$S_LASTCOLL        = 00000008
MCA$S_MCA             = 0000003A
MCA$V_ENTRY           = 00000000
MCA$V_EOF             = 00000003
MCA$V_ERA_SCRL        = 00000006
MCA$V_FILLER          = 0000000A
MCA$V_FUTURE          = 00000001
MCA$V_GRAPHICS        = 00000005
MCA$V_MULTFND         = 00000002
MCA$V_REFRESH         = 00000008
MCA$V_S_TOP_DISP      = 00000009
MCA$V_TOP_DISP        = 00000007
MCA$V_VIDEO           = 00000004
MCA$W_DCLASSCT        = 00000038
MCA$W_FLAGS           = 00000032
MCAPTR                ******** X     03
MIN_STAT              = 00000003
MMG$GL_NPAGEDYN       ******** X     03
MMG$GL_NPAGNEXT       ******** X     03
MMG$GL_SYSPHD         ******** X     03
MNR_CLS$B_TYPE        = 00000000
MNR_CLS$K_HSIZE       = 0000000D
MNR_CLS$Q_STAMP       = 00000003
MNR_CLS$S_CLASS_HDR   = 0000000D
MNR_CLS$S_FILLER      = 0000000F
MNR_CLS$S_FLAGS       = 00000002
```

```
MNR_CLS$S_STAMP                 = 00000008
MNR_CLS$V_CONT                  = 00000000
MNR_CLS$V_FILLER                = 00000001
MNR_CLS$W_FLAGS                 = 00000001
MNR_CLS$W_RESERVED              = 00000008
MNR_HDR$B_TYPE                  = 00000000
MNR_HDR$K_CLASSBITS             = 00000073
MNR_HDR$K_MAXCOMLEN             = 0000003C
MNR_HDR$K_REVLEVELS             = 00000083
MNR_HDR$K_SIZE                  = 00000103
MNR_HDR$L_FLAGS                 = 00000001
MNR_HDR$L_INTERVAL              = 00000015
MNR_HDR$L_RECCT                 = 00000029
MNR_HDR$O_CLASSBITS             = 00000073
MNR_HDR$O_REVOCLSBITS           = 00000019
MNR_HDR$Q_BEGINNING             = 00000005
MNR_HDR$Q_ENDING                = 0000000D
MNR_HDR$S_BEGINNING             = 00000008
MNR_HDR$S_CLASSBITS             = 00000010
MNR_HDR$S_COMMENT               = 0000003C
MNR_HDR$S_ENDING                = 00000008
MNR_HDR$S_FILE_HDR              = 00000103
MNR_HDR$S_FILLER                = 00000020
MNR_HDR$S_FLAGS                 = 00000004
MNR_HDR$S_LEVEL                 = 00000008
MNR_HDR$S_REVOCLSBITS           = 00000010
MNR_HDR$S_REVLEVELS             = 00000080
MNR_HDR$S_TYPE                  = 00000008
MNR_HDR$T_COMMENT               = 00000035
MNR_HDR$T_LEVEL                 = 0000002D
MNR_HDR$T_REVLEVELS             = 00000083
MNR_HDR$V_FILLER                = 00000000
MNR_HDR$W_COMLEN                = 00000071
MNR_HOM$K_PSIZE                 = 00000008
MNR_HOM$L_ELTCT                 = 00000000
MNR_HOM$L_RESERVED              = 00000004
MNR_HOM$S_HOM_CLASS_PRE         = 00000008
MNR_PRO$B_PRI                   = 0000000A
MNR_PRO$K_DSIZE                 = 0000003B
MNR_PRO$K_FSIZE                 = 00000040
MNR_PRO$K_PSIZE                 = 00000008
MNR_PRO$K_REVODSIZE             = 00000033
MNR_PRO$K_REV1DSIZE             = 0000003B
MNR_PRO$L_BIOCNT                = 0000002F
MNR_PRO$L_CPUTIM                = 00000028
MNR_PRO$L_DIOCNT                = 00000023
MNR_PRO$L_EFWM                  = 00000037
MNR_PRO$L_EPID                  = 00000033
MNR_PRO$L_IPID                  = 00000000
MNR_PRO$L_PAGEFLTS              = 00000027
MNR_PRO$L_PCTINT                = 00000004
MNR_PRO$L_PCTREC                = 00000000
MNR_PRO$L_STS                   = 0000001F
MNR_PRO$L_UIC                   = 00000004
MNR_PRO$O_LNAME                 = 00000008
MNR_PRO$S_LNAME                 = 00000010
MNR_PRO$S_PROCESS_CLASS         = 0000003B
MNR_PRO$S_PRO_CLASS_PRE         = 00000008
MNR_PRO$W_GPGCNT                = 0000001B
MNR_PRO$W_PPGCNT                = 0000001D
MNR_PRO$W_STATE                 = 00000008
MNR_SCS$C_CBLENGTH              = 00000038
MNR_SCS$C_CBWORK                = 00000044
MNR_SCS$L_CBKBMAPD              = 00000040
MNR_SCS$L_CBKBREQD              = 0000003C
MNR_SCS$L_CBKBSENT              = 00000038
MNR_SCS$L_DGDISCARD             = 00000010
MNR_SCS$L_DGRCVD                = 0000000C
MNR_SCS$L_DGSENT                = 00000008
MNR_SCS$L_KBYTMAPD              = 0000002C
MNR_SCS$L_KBYTREQD              = 00000028
MNR_SCS$L_KBYTSENT              = 00000020
MNR_SCS$L_MSGRCVD               = 00000018
MNR_SCS$L_MSGSENT               = 00000014
MNR_SCS$L_QBDT_CNT              = 00000034
MNR_SCS$L_QCR_CNT               = 00000030
MNR_SCS$L_REQDATS               = 00000024
MNR_SCS$L_SNDDATS               = 0000001C
MNR_SCS$Q_NODENAME              = 00000000
MNR_SYI$B_MPCPUS                = 0000000D
MNR_SYI$B_TYPE                  = 00000000
MNR_SYI$K_BALSETMEM             = 0000001E
MNR_SYI$K_CPUTYPE               = 00000026
MNR_SYI$K_MPWHILIM              = 00000022
MNR_SYI$K_NODENAME              = 0000000E
MNR_SYI$K_SIZE                  = 0000002A
MNR_SYI$L_BALSETMEM             = 0000001E
MNR_SYI$L_CPUTYPE               = 00000026
MNR_SYI$L_MPWHILIM              = 00000022
MNR_SYI$Q_BOOTTIME              = 00000003
MNR_SYI$S_BOOTTIME              = 00000008
MNR_SYI$S_FILLER                = 0000000E
MNR_SYI$S_FLAGS                 = 00000002
MNR_SYI$S_NODENAME              = 00000010
MNR_SYI$S_SYS_INFO              = 0000002A
MNR_SYI$S_TYPE                  = 00000008
MNR_SYI$T_NODENAME              = 0000000E
MNR_SYI$V_CLUSMEM               = 00000000
MNR_SYI$V_FILLER                = 00000002
MNR_SYI$V_RESERVED1             = 00000001
MNR_SYI$W_FLAGS                 = 00000001
MNR_SYI$W_MAXPRCCT              = 0000000B
MODES_PRE                         00000361 RG      03
MPS$AC_CPUTIME                    ******** X       03
MPS$GL_NULLCPU                    ******** X       03
MPS$GQ_MPSTRTIM                   ******** X       03
MPSTRTIM                          0000006C R       01
MRB                             = 00000000
MRB$A_COMMENT                   = 0000002C
MRB$A_DISPLAY                   = 00000020
MRB$A_INPUT                     = 0000001C
MRB$A_RECORD                    = 00000024
MRB$A_SUMMARY                   = 00000028
MRB$B_INP_FILES                 = 00000042
```

```
MRB$K_SIZE                      = 00000045
MRB$L_FLUSH                     = 00000014
MRB$L_INTERVAL                  = 00000010
MRB$L_VIEWING_TIME              = 00000018
MRB$M_ALL_CLASS                 = 00000400
MRB$M_BY_NODE                   = 00001000
MRB$M_DISPLAY                   = 00000001
MRB$M_DISP_TO_FILE              = 00000020
MRB$M_DIS_CL_REQ                = 00000100
MRB$M_INDEFEND                  = 00000010
MRB$M_INP_CL_REQ                = 00000040
MRB$M_MFSDM                     = 00000800
MRB$M_PLAYBACK                  = 00000008
MRB$M_PROC_REQ                  = 00004000
MRB$M_RECORD                    = 00000002
MRB$M_REC_CL_REQ                = 00000080
MRB$M_SUMMARY                   = 00000004
MRB$M_SUM_CL_REQ                = 00000200
MRB$M_SYSCLS                    = 00002000
MRB$O_CLASSBITS                 = 00000032
MRB$Q_BEGINNING                 = 00000000
MRB$Q_ENDING                    = 00000008
MRB$S_BEGINNING                 = 00000008
MRB$S_CLASSBITS                 = 00000010
MRB$S_ENDING                    = 00000008
MRB$S_FLAGS                     = 00000002
MRB$S_MRB                       = 00000045
MRB$V_ALL_CLASS                 = 0000000A
MRB$V_BY_NODE                   = 0000000C
MRB$V_DISPLAY                   = 00000000
MRB$V_DISP_TO_FILE              = 00000005
MRB$V_DIS_CL_REQ                = 00000008
MRB$V_FILLER                    = 0000000F
MRB$V_INDEFEND                  = 00000004
MRB$V_INP_CL_REQ                = 00000006
MRB$V_MFSDM                     = 0000000B
MRB$V_PLAYBACK                  = 00000003
MRB$V_PROC_REQ                  = 0000000E
MRB$V_RECORD                    = 00000001
MRB$V_REC_CL_REQ                = 00000007
MRB$V_SUMMARY                   = 00000002
MRB$V_SUM_CL_REQ                = 00000009
MRB$V_SYSCLS                    = 0000000D
MRB$W_CLASSCT                   = 00000030
MRB$W_FLAGS                     = 00000043
NO                                ********    X    03
OTHER_STATES                      00000094 RG      01
PAGE_PRE                          000002AF RG      03
PB$L_CDTLST                     = 00000034
PB$L_FLINK                      = 00000000
PCB$B_PRI                       = 0000000B
PCB$L_EFWM                      = 0000004C
PCB$L_EPID                      = 00000064
PCB$L_PHD                       = 0000006C
PCB$L_PID                       = 00000060
PCB$L_STS                       = 00000024
PCB$L_UIC                       = 000000BC
PCB$T_LNAME                     = 00000070
PCB$V_RES                       = 00000000
PCB$W_GPGCNT                    = 00000034
PCB$W_PPGCNT                    = 00000036
PCB$W_STATE                     = 0000002C
PHD$L_BIOCNT                    = 00000058
PHD$L_CPUTIM                    = 00000038
PHD$L_DIOCNT                    = 00000054
PHD$L_PAGEFLTS                  = 0000004C
PMS$GL_BLK_IN                     ********    X    03
PMS$GL_BLK_LOC                    ********    X    03
PMS$GL_BLK_OUT                    ********    X    03
PMS$GL_DEQ_IN                     ********    X    03
PMS$GL_DEQ_LOC                    ********    X    03
PMS$GL_DEQ_OUT                    ********    X    03
PMS$GL_DIRDATA_HIT                ********    X    03
PMS$GL_DIRDATA_MISS               ********    X    03
PMS$GL_DIRHIT                     ********    X    03
PMS$GL_DIRMISS                    ********    X    03
PMS$GL_DLCKMSGS_IN                ********    X    03
PMS$GL_DLCKMSGS_OUT               ********    X    03
PMS$GL_ENQCVT_IN                  ********    X    03
PMS$GL_ENQCVT_LOC                 ********    X    03
PMS$GL_ENQCVT_OUT                 ********    X    03
PMS$GL_ENQNEW_IN                  ********    X    03
PMS$GL_ENQNEW_LOC                 ********    X    03
PMS$GL_ENQNEW_OUT                 ********    X    03
PMS$GL_EXTHIT                     ********    X    03
PMS$GL_EXTMISS                    ********    X    03
PMS$GL_FCP2                       ********    X    03
PMS$GL_FIDHIT                     ********    X    03
PMS$GL_FIDMISS                    ********    X    03
PMS$GL_FILHDR_HIT                 ********    X    03
PMS$GL_FILHDR_MISS                ********    X    03
PMS$GL_KERNEL                     ********    X    03
PMS$GL_QUOHIT                     ********    X    03
PMS$GL_QUOMISS                    ********    X    03
PMS$GL_STORAGMAP_HIT              ********    X    03
PMS$GL_STORAGMAP_MISS             ********    X    03
POOL_PRE                          00000069 RG      03
PR$_IPL                         = 00000012
PROCDISPS                       = 00000005
PROCESS_CLASS                   = 00000000
PROC_COUNT                        00000090 RG      01
PROC_PRE                          000004A5 RG      03
PRO_CLASS_PRE                   = 00000000
QUAL$A_ALL                      = 00000064
QUAL$A_AVE                      = 00000074
QUAL$A_BEG                      = 00000004
QUAL$A_BY_NODE                  = 00000054
QUAL$A_CLASS                    = 0000005C
QUAL$A_COMM                     = 0000004C
QUAL$A_CPU                      = 000000AC
QUAL$A_CUR                      = 0000006C
QUAL$A_DISP                     = 00000034
QUAL$A_END                      = 0000000C
QUAL$A_FLUSH                    = 0000001C
```

```
QUAL$A_INP                      = 0000002C
QUAL$A_INT                      = 00000014
QUAL$A_ITEM                     = 000000BC
QUAL$A_MAX                      = 00000084
QUAL$A_MIN                      = 0000007C
QUAL$A_PCENT                    = 000000B4
QUAL$A_REC                      = 0000003C
QUAL$A_SUMM                     = 00000044
QUAL$A_TOPB                     = 0000009C
QUAL$A_TOPC                     = 0000008C
QUAL$A_TOPD                     = 00000094
QUAL$A_TOPF                     = 000000A4
QUAL$A_VIEW                     = 00000024
QUAL$L_ALL                      = 00000060
QUAL$L_AVE                      = 00000070
QUAL$L_BEG                      = 00000000
QUAL$L_BY_NODE                  = 00000050
QUAL$L_CLASS                    = 00000058
QUAL$L_COMM                     = 00000048
QUAL$L_CPU                      = 000000A8
QUAL$L_CUR                      = 00000068
QUAL$L_DISP                     = 00000030
QUAL$L_END                      = 00000008
QUAL$L_FLUSH                    = 00000018
QUAL$L_INP                      = 00000028
QUAL$L_INT                      = 00000010
QUAL$L_ITEM                     = 000000B8
QUAL$L_MAX                      = 00000080
QUAL$L_MIN                      = 00000078
QUAL$L_PCENT                    = 000000B0
QUAL$L_REC                      = 00000038
QUAL$L_SUMM                     = 00000040
QUAL$L_TOPB                     = 00000098
QUAL$L_TOPC                     = 00000088
QUAL$L_TOPD                     = 00000090
QUAL$L_TOPF                     = 000000A0
QUAL$L_VIEW                     = 00000020
QUAL$S_QUALIFIER_DESC           = 000000C0
QUALIFIER_DESC                  = 00000000
QUO_TRIES                       000000B4 RG     01
REG_PROC                        = 00000000
RESCNT                          00000060 RG     01
SB$L_FLINK                      = 00000000
SB$L_PBFL                       = 0000000C
SB$T_NODENAME                   = 00000044
SB$T_SWTYPE                     = 00000024
SCANDISKS                       000005B6 R      03
SCANJDEVICES                    00000675 R      03
SCANLRP                         0000027C R      03
SCANPOOL                        00000086 R      03
SCANPROCS                       000004C2 R      03
SCANSCS                         000007C0 R      03
SCH$C_COM                       = 0000000C
SCH$C_COMO                      = 0000000D
SCH$C_HIB                       = 00000007
SCH$C_HIBO                      = 00000008
SCH$C_LEF                       = 00000005
SCH$C_LEFO                      = 00000006
SCH$C_MWAIT                     = 00000002
SCH$C_PFW                       = 00000004
SCH$GL_CURPCB                   ********   X   03
SCH$GL_MAXPIX                   ********   X   03
SCH$GL_PCBVEC                   ********   X   03
SCH$IOLOCKR                     ********   X   03
SCH$IOUNLOCK                    ********   X   03
SCS$GA_LOCALSB                  ********   X   03
SCS$GQ_CONFIG                   ********   X   03
SCS_PRE                         000007AA RG     03
SMALCNT                         00000024 RG     01
SMALLHOLE                       00000028 RG     01
SPTR                            ********    X   03
SRPCNT                          00000034 RG     01
SRPINUSE                        00000038 RG     01
SS$_NORMAL                      ********    X   03
STATES_PRE                      000002CD RG     03
STATS                           = 00000005
STORAGMAP_TRIES                 000000B8 RG     01
SYS$CMKRNL                      ********  GX    03
SYS$LKWSET                      ********  GX    03
SYS$ULWSET                      ********  GX    03
SYSFAULTS                       00000048 RG     01
SYSMGR_STATES                   00000098 R      01
SYSMGR_STATETOT                 = 00000008
SYS_INFO                        = 00000000
TOPB_PROC                       = 00000003
TOPC_PROC                       = 00000001
TOPD_PROC                       = 00000002
TOPF_PROC                       = 00000004
UCB$B_DEVCLASS                  = 00000040
UCB$B_FIPL                      = 0000000B
UCB$L_DEVCHAR                   = 00000038
UCB$L_DEVCHAR2                  = 0000003C
UCB$L_FQFL                      = 00000000
UCB$L_JNL_BWCNT                 = 000000EC
UCB$L_JNL_EXCNT                 = 000000F0
UCB$L_JNL_FQFL                  = 000000B0
UCB$L_JNL_WQFL                  = 000000A8
UCB$L_JNL_WRCNT                 = 000000E8
UCB$L_OPCNT                     = 00000070
UCB$L_VCB                       = 00000034
UCB$W_QLEN                      = 0000006A
UCB$W_UNIT                      = 00000054
VCB$T_VOLNAME                   = 00000014
YES                             ********   X   03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
DSPDATA	000000C0 (192.)	01 (1.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC QUAD
ABS	00000000 (0.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$MONCODE	00000987 (2439.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.08	00:00:00.80
Command processing	129	00:00:00.68	00:00:04.90
Pass 1	502	00:00:19.38	00:00:49.82
Symbol table sort	0	00:00:03.20	00:00:05.56
Pass 2	369	00:00:06.15	00:00:14.16
Symbol table output	41	00:00:00.46	00:00:01.43
Psect synopsis output	0	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1075	00:00:29.99	00:01:16.83

The working set limit was 2400 pages.
117204 bytes (229 pages) of virtual memory were used to buffer the intermediate code.
There were 110 pages of symbol table space allocated to hold 1976 non-local and 89 local symbols.
2238 source lines were read in Pass 1, producing 59 object records in Pass 2.
46 pages of virtual memory were used to define 34 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[MONTOR.OBJ]MONLIB.MLB;1	5
_255DUA28:[SYS.OBJ]LIB.MLB;1	14
_255DUA28:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	30

2010 GETS were required to define 30 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:PREPOST/OBJ=OBJ$:PREPOST MSRC$:PREPOST/UPDATE=(ENH$:PREPOST)+EXECML$/LIB+LIB$:MONLIB/LIB
```